Technical Report

UCAM-CL-TR-895
ISSN 1476-2986

Number 895

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Artificial error generation for translation-based grammatical error correction

Mariano Felice

October 2016

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 October 2016 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Hughes Hall.

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

Artificial error generation for translation-based grammatical error correction

Mariano Felice

Summary

Automated grammatical error correction for language learners has attracted a lot of attention in recent years, especially after a number of shared tasks that have encouraged research in the area. Treating the problem as a translation task from 'incorrect' into 'correct' English using statistical machine translation has emerged as a state-of-the-art approach but it requires vast amounts of corrected parallel data to produce useful results. Because manual annotation of incorrect text is laborious and expensive, we can generate artificial error-annotated data by injecting errors deliberately into correct text and thus produce larger amounts of parallel data with much less effort.

In this work, we review previous work on artificial error generation and investigate new approaches using random and probabilistic methods for constrained and general error correction. Our methods use error statistics from a reference corpus of learner writing to generate errors in native text that look realistic and plausible in context. We investigate a number of aspects that can play a part in the error generation process, such as the origin of the native texts, the amount of context used to find suitable insertion points, the type of information encoded by the error patterns and the output error distribution. In addition, we explore the use of linguistic information for characterising errors and train systems using different combinations of real and artificial data.

Results of our experiments show that the use of artificial errors can improve system performance when they are used in combination with real learner errors, in line with previous research. These improvements are observed for both constrained and general correction, for which probabilistic methods produce the best results. We also demonstrate that systems trained on a combination of real and artificial errors can beat other highly-engineered systems and be more robust, showing that performance can be improved by focusing on the data rather than tuning system parameters.

Part of our work is also devoted to the proposal of the I-measure, a new evaluation scheme that scores corrections in terms of improvement on the original text and solves known issues with existing evaluation measures.

To my family and L.A.
In memory of Kika.

Acknowledgements

This piece of work is the culmination of a very intense period of my life, both on an academic and personal level. Reaching this far would not have been possible without the help and support of many people, who deserve to be mentioned in this short but very important page.

First and foremost, I am especially grateful to my family, who have always supported me in many different ways and given me the energy to go on. Everything I do will always be for them. I am equally grateful to my dear L.A. for always being there for me, whatever I do and wherever I go. You are a very important part of my life. No more thesis now!

I am of course deeply indebted to my supervisor, Prof Ted Briscoe, for believing in me and giving me very valuable advice and freedom to mature as a researcher. I extend my gratitude to all the members of the ALTA Institute at the Computer Laboratory for always being willing to share their knowledge with me and give me a hand whenever I needed it. I am also particularly grateful my fellow PhD students in the office: thank you for your kindness and all the group therapy! Special thanks are due to Zheng Yuan for her constant support, endless fruitful discussions and disinterested help; you are a very dear friend. My gratitude goes also to Christopher Bryant for his generous revision of my thesis draft.

Thanks are also due to my examiners, Dr Paula Buttery and Prof Stephen Pulman, for making my viva a very pleasant experience and providing me with very valuable feedback.

Last, but not least, I would like to thank Cambridge English Language Assessment for giving me the opportunity to pursue a degree in one of the most prestigious universities in the world while I worked a topic that I really enjoyed. I am also grateful to the Computer Laboratory for providing me with additional funding in the last stage of my studies as well as to my college, Hughes Hall, for their financial support to attend scientific meetings.

Contents

List of Abbreviations

Acc	accuracy
AEG	artificial error generation
BLEU	Bilingual Evaluation Understudy
BNC	British National Corpus
CEFR	Common European Framework of Reference for Languages
CLC	Cambridge Learner Corpus
CoNLL	Conference on Natural Language Learning
EFCamDat	EF-Cambridge Open Language Database
ESOL	English as a Second or Other Language
EVP	English Vocabulary Profile
F	F-measure
FCE	First Certificate in English
FN	false negative
FP	false positive
GEC	grammatical error correction
GLEU	Generalized Language Evaluation Understanding
HMM	Hidden Markov Model
HOO	Helping Our Own
I	I-measure
JSD	Jensen-Shannon Divergence
KLD	Kullback–Leibler Divergence
L1	first language
L2	second language
LLC	Longman Learners' Corpus
LM	language model
ML	machine learning
MT	machine translation
NLP	Natural Language Processing
NUCLE	National University of Singapore Corpus of Learner English
P	precision
PoS	part-of-speech
R	recall

RASP	Robust Accurate Statistical Parser
SLA	Second Language Acquisition
SMT	Statistical Machine Translation
SP	Sum of Pairs
TN	true negative
TP	true positive
WAcc	weighted accuracy
WAS	Writer-Annotator-System

Chapter 1

Introduction

This thesis explores the use of techniques from the field of Natural Language Processing (NLP) to aid written production in English as a Second or Other Language (ESOL). In this introductory chapter, we review the importance of English as a global language, the challenges for learners of the language and how technology fits into the equation. The end of the chapter sets out our research objectives and gives an outline of this dissertation.

1.1 English in the modern world

The English language has grown to become a *lingua franca*, a common global language for business, work, education and research, which also permeates other aspects of our lives. It is estimated that the number of speakers who have English as their native language (or L1) is over 335 million whereas those who speak it as a second language (or L2) account for 505 million (Lewis et al., 2015). However, depending on where we draw the boundaries, the number of people who speak English at a 'useful level' can go up to 1.75 billion (Howson, 2013) or even as high as 2 billion (Crystal, 2008). According to Crystal, for every one native speaker, there are three or four non-native speakers, a ratio which is likely to increase.

The growing interest in English is also attested by the number of candidates sitting recognised international examinations. Annual candidature for Cambridge English's main suite exams was over 1 million by 2002, over 2 million by 2007 and more than 4 million by 2013 (Hawkey and Milanovic, 2013). These figures translate into a clear demand for courses and resources involved in foreign language teaching. In the last decades, language learning has evolved from a rather 'static' drill-centred activity to an interactive experience focused on communication skills. The advancement of technology has also opened up new possibilities, ranging from distance learning to readily available online tools. It is generally believed that if we can decipher how humans learn and process language, we will be able to develop better technology to support the learning process.

1.2 Second Language Acquisition

Despite all the existing incentives and supporting material, learning a new language is not a trivial task, even less so when it is undertaken in a conscious manner. The field of Second Language Acquisition (SLA) seeks to discover and explain the way in which humans learn a new language after the native language has been learned.

Krashen (1982, p. 10) makes a crucial distinction between *acquisition* and *learning*. *Acquisition* is defined as 'a process similar, if not identical to the way children develop ability in their first language' and is therefore 'subconscious' whereas *learning* concerns the 'way to develop competence in a second language', the 'conscious knowledge of a second language'. Since the new language often comes after a first language has solidified, individuals are likely to transfer forms and meanings of their native language and culture to the foreign language and culture (Lado, 1957, p. 2). Thus, in early stages of learning, a native Spanish speaker is likely to produce **I am lawyer* instead of the correct *I am a lawyer*.[1]

L1 influence is a well-known source of errors in non-native writing. Several studies have shown that the errors induced by a learner's native language are systematic and can usually be predicted given their L1 (Berzak et al., 2015; Gass and Selinker, 1992; Swan and Smith, 2001). For example, learners whose L1 does not have an article system (such as Russian or Japanese) tend to have problems mastering articles in English. It is also possible to infer a person's L1 based solely on a sample of their writing, a task known as *native language identification*. Its potential use in forensic linguistics and tailored error feedback led to the organisation of a shared task in 2013, with 29 competing systems achieving a maximum overall accuracy of over 83% (Tetreault et al., 2013).

Knowing a learner's L1 can also be used to improve grammatical error correction (GEC) systems. In particular, Rozovskaya and Roth (2011) have shown that using L1-specific correction probabilities as priors for Bayesian approaches can improve system performance and provide L1-adapted models without having to train a new system for each native language .

1.3 Grammatical errors

Consciously or not, L2 learners progressively internalise the grammar of the target language in order to produce well-formed sentences and communicate with others. It is beyond the scope of this thesis to discuss whether grammar should be taught explicitly or not but, given the scope of our research, we will assume it is still a matter of concern for L2 learners and teachers. Learners aim to be more proficient in their target language and teachers work towards helping them achieve that goal.

[1] By convention, ungrammatical constructions are preceded by an asterisk (*).

What constitutes an error in an L2-learning environment varies across authors and periods. Corder (1967), cited by Ellis (2008, p. 961), describes an *error* as 'a deviation in learner language which results from lack of knowledge of the correct rule'. For Gass and Selinker (1994, p. 66) errors are 'red flags' that 'provide evidence of the state of a learner's knowledge of the L2' while for James (1998, p. 78) an error is 'an instance of language that is unintentionally deviant and not self-corrigible by its author'.

A distinction is often made between *errors* and *mistakes* in the SLA literature. Ellis (2008, p. 971) defines Corder's concept of *mistake* as 'a deviation in learner language that occurs when learners fail to perform their competence', 'a lapse that reflects processing problems'. According to James (1998, p. 83), errors cannot be self-corrected but mistakes can if the deviance is pointed out to the learner. As Brown (2014, p. 249) explains, an error is 'a noticeable deviation from the adult grammar of a native speaker' that 'reflects the competence of the learner'. For example, **Does John can sing?* is likely to indicate incomplete knowledge of auxiliary verbs for question formation. A mistake, on the other hand, is 'a random guess or a slip', 'a failure to utilize a known system correctly'. As Brown points out, all people make mistakes, including native speakers. A common example is the inadvertent use of a homophone instead of the intended word, such as *there* for *their*.

Although the error/mistake distinction seems crucial in SLA, it has been regarded as purely 'academic' and not relevant for teachers (Bartram and Walton, 1991, p. 20). A more comprehensive and practical definition is given by Ferris (2011, p. 3): 'Errors are morphological, syntactic, and lexical forms that deviate from rules of the target language, violating the expectations of literate adult native speakers'. This definition is closer to the notion of grammatical error used in NLP, which includes morphology, syntax and mechanical errors but largely excludes usage and spelling mistakes (Leacock et al., 2014, p. 2). In fact, only spelling errors which constitute a breach of syntactic or morphological rules are of interest for grammatical error correction. Typical examples include wrong word forms (**breaked* instead of *broke*), confusion between homophones (*it's–its*) and typographical errors that result in other valid words which are wrong in the given context (*sing–sign*).

1.4 Automatic error detection and correction

Research on automatic GEC is based on the assumption that correcting errors is beneficial for learners. However, there does not seem to be a general consensus on this matter. In a controversial essay, Truscott (1996) claimed that error correction was ineffective and harmful to second language learning and should be abandoned. Although a number of studies are in line with his views (Cohen and Robbins, 1976; Polio et al., 1998; Truscott and Hsu, 2008), most researchers in the field of second-language learning rebutted such claims, proving that error feedback improves writing

accuracy either in the short or long term (Ashwell, 2000; Bitchener and Knoch, 2008, 2010; Ellis et al., 2008; Ferris, 2006; Sheen, 2007; van Beuningen et al., 2012). Similar studies have also shown that automated feedback is helpful in reducing error rates for both native and non-native speakers, highlighting the value of GEC systems as writing assistants (Andersen et al., 2013; Attali, 2004; Chodorow et al., 2010; Choi and Lee, 2010; Lavolette et al., 2015; Lipnevich and Smith, 2008; Nagata and Nakatani, 2010; Shermis et al., 2008).

Corrective feedback can be *direct*, when the correct linguistic form is provided to the students, or *indirect*, when they are told that an error has been made but it is up to them to correct it (Ferris, 2011, p. 31). These notions are closely related to two distinct subtasks within automated GEC: error *detection* and error *correction*. Detection is only concerned with identifying pieces of text that are grammatically incorrect. This can range from a binary output (e.g. whether a sentence is incorrect or not) to the specific location and span of an error, including optional information such as a type or a description. Correction, on the other hand, refers to the actual suggestions given to fix an error (i.e. direct feedback), sometimes linked to explanations or further examples.

Detection and correction are two different tasks but they are not always two separate processes. Although it is generally understood that detection is a necessary step before correction, many systems are aimed directly at correction with no explicit detection, such as systems based on Statistical Machine Translation (SMT).

Just as there is debate about whether indirect feedback is better than direct feedback (Ferris and Hedgcock, 2014, p. 287), there is also the question of whether detection is actually more useful than correction. In fact, it may be easier for systems to detect something that is incorrect rather than generate a suitable correction, especially when errors are complex or systems are not confident about their corrections. In such cases, detection alone may be preferred, as it is generally believed that learners are more likely to benefit from a revision hint than from a wrong (and therefore misleading) correction.

1.4.1 Challenges

There are a number of reasons why the error correction task is particularly challenging. First of all, both native and non-native speakers write mostly error-free text, with errors being the exception rather than the rule. For example, articles are used incorrectly only about 10%–13% of the times among intermediate learners of English (Han et al., 2006; Rozovskaya and Roth, 2010c), setting a very high baseline for the task. In such circumstances, using a system with less than 90% accuracy will be worse than leaving the text uncorrected, whereupon 'doing nothing' will remain the wisest strategy.

Identifying error types in L2 writing is also crucial for the development of successful error correction systems. Although there is some overlap between native and non-native errors (e.g. misusing punctuation or confusing homophones), L2-learners commit other types of errors that are rather infrequent among native speakers, usually caused by the lack of clear-cut usage rules (e.g. collocations) or L1 grammar transfers (Leacock et al., 2014, p. 17). In addition, correctness is sometimes judged solely on the native language of the speaker, so the same utterance can be deemed correct for native speakers but incorrect for L2-speakers. This is often the case for creative use of language, where licence is only given to L1-speakers. As Prodromou (2008, p. 232) puts it, 'the language is mistake-proof *for the L1-user*' [original emphasis].

Different types of errors require different correction strategies so it is often very difficult to integrate them into a single system. The complexity of integration is twofold. Firstly, there is the intrinsic complexity of each error type. Some errors are easier to correct than others and this often corresponds with the number of possible corrections (Bryant and Ng, 2015). Closed-class errors are limited to a predefined set of alternatives (or *confusion set*) so they are, in principle, easier to deal with. Articles are the most common example, where there are only four possible alternatives: no article (Ø), *a*, *an* or *the*. Open-class errors involving verbs, adjectives and nouns (such as collocations) operate on a much wider search space and thus carry higher chances of introducing new mistakes if a wrong choice is made. This is one of the reasons why most successful systems focus only on a limited number of error types, especially those with closed-class sets such as articles, prepositions and verb tenses.

Secondly, there is the complexity of interacting errors or how the correction of one error affects others. Some errors might be corrected in isolation while others might depend on previous corrections. Combining corrections for all the errors in a sentence is not a trivial problem. Even the pipelined solutions that correct one type after the other need to figure out the optimal correction pipeline, which is not always possible when there is one component per error type for a large typology (e.g. as few as 10 error types would yield $10! = 3,628,800$ possible pipelines). In reality, the order in which errors are corrected depends on each sentence and it would be wrong to assume that a deterministic type-after-type pipeline is the optimal solution to all cases.

The number of errors in a sentence can also affect system performance. As the number of errors in a sentence increases, so does correction complexity, since the system must ensure all the corrections are compatible in the output. The co-occurrence of many errors in a single sentence also means that the reliable context on which a system bases its decisions is reduced, leading to lower confidence and precision. This raises some fundamental questions about error correction: To what extent can we correct a sentence? What is the maximum number of errors that make a sentence worth correcting? Are some sentences beyond repair? When should we discard a sentence completely and rewrite it from scratch?

Finally, given the infinite nature of language, it is impossible for a system to know all possible well-formed sentences and make perfect judgements. Similarly, there could be a potentially innumerable set of acceptable corrections for a given error, so determining whether a correction is valid or not is almost as difficult as correction itself. This is particularly important for system evaluation, where a given output might not match the expected corrections but still provide a valid alternative.

1.5 Research goals

In the last few years, SMT has emerged as a powerful approach for general error correction, achieving state-of-the-art performance over more traditional approaches such as classifiers and language models. However, the success of SMT models depends largely on the quantity and quality of training data (Gavrila and Vertan, 2011; Koehn et al., 2003; Suresh, 2010), which is currently very limited for this task. A few proposals have tried to overcome this data scarcity problem by injecting artificial errors into error-free text, thereby generating vast amounts of data with minimum effort.

The quality and impact of artificial data can be affected by a number of factors, such as the type of texts, errors and methods used for generation. In this thesis, we set out to investigate different aspects of artificial error generation and how they affect the performance of SMT-based GEC systems.

Our research objectives can be summarised as follows:

1. Exploit the potential of SMT systems for general error correction, not limited to most common error types.

2. Find out whether system performance can be improved by focusing on data rather than system engineering.

3. Analyse the variables involved in artificial error generation, discovering the optimal settings that lead to higher quality data and better system performance.

4. Investigate different artificial error generation methods and recommend a framework for the generation of artificial data for GEC purposes.

5. Review existing evaluation methods in the field and propose a new evaluation scheme to overcome their limitations.

It is not the aim of this thesis to provide an in-depth study of SMT for GEC. For this reason, we do not address issues such as optimal data structures, alignment, decoding algorithms, tuning strategies or other technical aspects that are not explicitly mentioned in our experiments. SMT is only used as a common experimental framework for testing the performance of our artificial datasets, since it allows us to build state-of-the-art GEC systems with minimal effort.

1.6 Thesis structure

The remainder of this thesis is structured as follows. Chapter 2 describes related work on GEC. In Section 2.1, we give a historical overview of work in the field, from early GEC systems to state-of-the-art data-driven approaches. Special attention is given to the use of SMT systems for error correction purposes, since this is the approach adopted for our experiments. In Section 2.2 we describe the type of data that is typically used to build such systems, including native text, error-annotated learner corpora and artificially-generated data. Here, we introduce the notion of 'artificial errors', which form the cornerstone of this thesis. Finally, Section 2.3 describes a number of organised competitions (or 'shared tasks') that have provided a common environment for system development and testing, and have greatly contributed to the advancement of research in the area.

Chapter 3 addresses the issue of system evaluation for GEC. In Section 3.1 we review the most common approaches, from traditional evaluation measures to specific scorers designed for the shared tasks. Given a number of limitations of these methods, in Section 3.2 we propose the I-measure (Felice and Briscoe, 2015), a new evaluation scheme to score corrections in terms of improvement over the original uncorrected text. A few more recent approaches using crowdsourcing for system evaluation are reviewed in Section 3.3 and a discussion of a suitable method for the computation of statistical significance is included in Section 3.4. The chapter concludes with a few considerations in Section 3.5.

Chapter 4 describes our first set of experiments, aimed at correcting only five types of errors concerning articles and determiners, noun number, prepositions, subject-verb agreement and verb forms. In Section 4.1 we explain our motivations for investigating artificial errors and review some of its challenges. The following sections present experiments on artificial error generation using the CoNLL-2013 shared task datasets. Section 4.2 describes a random method that considers all errors to be equally probable while Section 4.3 presents a probabilistic method that preserves the original error frequencies. An analysis of the performance of both methods is given in Section 4.4, including a comparison with participating systems in the CoNLL-2013 shared task.

Chapter 5 describes our second set of experiments, aimed at general error correction (i.e. all types) using refined probabilistic methods. In Section 5.1 we describe our experimental set-up, such as the training corpus, error generation method and artificial datasets. Section 5.2 discusses the results of our experiments and provides a separate analysis for each tested variable: base texts, context window length, pattern type, generation mode and dataset size. Section 5.3 includes a comparison between our systems and the submissions to the CoNLL-2014 shared task.

Chapter 6 presents our conclusions and discusses directions for future work.

Parts of this work have been previously published in the following articles, which are referenced accordingly:

- Z. Yuan and M. Felice (2013). Constrained Grammatical Error Correction using Statistical Machine Translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 52–61

- M. Felice and Z. Yuan (2014a). 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*. Gothenburg, Sweden: Association for Computational Linguistics, pp. 116–126

- M. Felice et al. (2014). Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 15–24

- M. Felice and Z. Yuan (2014b). To Err is Human, to Correct is Divine. In *XRDS* 21 (1), pp. 22–27

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

Chapter 2

Background

This chapter gives a brief historical overview of research in GEC. The first section summarises the history of GEC systems, common approaches and the state of the art. The second section describes some of the most popular datasets in the field, including native, learner and artificial data. The final section gives an account of relevant shared tasks and their contributions to the field.

2.1 Grammatical error correction systems

2.1.1 Early approaches

The first grammar-checking systems date back to the 1980s and were essentially based on string-matching *rules*, such as The Writer's Workbench (Macdonald et al., 1982). Whenever a rule matches a given string in the text, the system flags up an error and provides a correction. However, rules could be triggered in inappropriate contexts and lead to false positives. For example, the system would suggest replacing *provided that* with *if* even if the phrase is not used with that meaning, as in *Joe provided that book and Mary this one.*

Other rule-based systems, such as Grammatik by Aspen Software and CorrecText by the Houghton Mifflin Company, incorporated some basic linguistic analysis and were eventually integrated into Microsoft Word and WordPerfect respectively in the early '90s. Among systems for language learners, ALEK (Chodorow and Leacock, 2000; Leacock and Chodorow, 2003), GRANSKA (Domeij et al., 2000) and ESL Assistant (Gamon et al., 2009) are prime examples based on hand-crafted heuristic rules.

Finally, the use of parsers and sophisticated grammars enabled systems to perform full syntactic analysis and identify grammatical violations using hand-coded rules (Arppe, 2000; Heidorn et al., 1982; Johannessen et al., 2002; Richardson and Braden-Harder, 1988). Typical examples include the identification of agreement errors and sentence fragments from a parser's output. Most, if not all, of today's grammar checking products employ this strategy, as it helps minimise the number of false

positives. Many open-source solutions, such as AbiWord[1], After the Deadline[2] and LanguageTool[3] (Naber, 2003), also rely on a linguistic analysis of the input and lists of error patterns, which are open to contributions from their users. Hand-crafted rules are also common in the NLP research community (Boroş et al., 2014; Bustamante and León, 1996; Ehsan and Faili, 2010; Flickinger and Yu, 2013; Gupta, 2014; Kunchukuttan et al., 2013; Lee and Lee, 2014; Sidorov et al., 2013; Wang et al., 2014a; Wu et al., 2014; Xiang et al., 2013), although they are often superseded by more powerful models. Apart from the quality of the rules themselves, much of the efficacy of rule-based systems depends on the robustness of automatic tools to process grammatically incorrect input (Leacock et al., 2014, p. 7).

2.1.2 Data-driven approaches

The emergence of statistical NLP in the late 1980s as a new way of dealing with language led to a growing interest in the compilation and exploitation of corpora. The *corpus-based* or *statistical* approaches to GEC use bodies of written text to extract linguistic knowledge or build models with machine learning (ML) techniques, as we explain the following sections.

2.1.2.1 Corpus-derived rules

As noted in Section 2.1.1, rules can be effective for error correction; however, the process of identifying and coding error patterns can be intellectually demanding and time consuming. Given an error-annotated corpus where incorrect phrases are mapped to their correct versions, it is possible to automatically extract patterns to identify and correct very frequent errors. For example, if we observe that the bigram **people is* is generally corrected to *people are*, we can create a rule for future correction. This can be done at different levels, such as words, part-of-speech (PoS) tags or dependency relations. However, care must be taken when extracting and applying such rules, since, in reality, corrections are not deterministic and might depend on larger context (e.g. *Recruiting the right people is essential for success* does not require correction).

The rule-extraction technique was first used to build spellcheckers. Mangu and Brill (1997) proposed a system that derived spelling-correction rules from the Brown corpus (Francis and Kucera, 1979) and a collection of confusion sets. Park and Rim (1998) and Byun et al. (2007) describe similar systems for Korean, although they use different extraction criteria.

Others have derived rules to correct hyphen usage. Rozovskaya et al. (2011) extracted mappings between hyphenated and non-hyphenated words from a corpus of scientific papers, comparing the frequency of observed token sequences as one

[1] http://www.abisource.com/
[2] http://www.afterthedeadline.com/
[3] http://www.languagetool.org/

word, with and without a hyphen. If one of these forms is at least 50% more frequent than the others, a correction rule is created. Similarly, Cahill et al. (2013a) describe two baselines for predicting missing hyphens, both based on patterns mined from a corpus. The first of them predicts a missing hyphen if a bigram occurs hyphenated more than 1,000 times in Wikipedia while the second system only does it when the probability of the hyphenated form is greater than 0.66.

For general error correction, the rule-based word-level component of the Self Assessment and Tutoring system (Andersen et al., 2013) uses patterns of unigrams, bigrams and trigrams extracted from the Cambridge Learner Corpus (CLC). In order to ensure high precision, the system only extracts n-grams which have been annotated as incorrect at least five times and ninety per cent of the times they occur. KNGED (Chang et al., 2014), a GEC system for Chinese, combines manual and automatically extracted rules based on words and PoS tags which are used to correct a broader range of error types.

2.1.2.2 Language models and n-gram counts

Language models (LMs) and n-gram frequency information are very useful for GEC, since frequency is generally viewed as a good proxy for well-formedness. These statistical representations of language have been used for different purposes in system development. Firstly, they seem naturally suited for error detection, since we can judge the grammaticality of a sentence or phrase from their estimated probabilities (Heilman et al., 2014; Hernandez and Calvo, 2014; Lee et al., 2014; Lin and Chen, 2015; Okanohara and Tsujii, 2007; Wagner et al., 2007).

Secondly, they can also be used for correction, either by predicting words in a given sequence or validating corrections (Bergsma et al., 2009; Islam and Inkpen, 2011; Kao et al., 2013; Lee and Lee, 2014; Wu et al., 2014; Xie et al., 2015; Zhang and Wang, 2014). In this case, correction candidates are taken from predefined sets (e.g. articles or prepositions) or generated 'on the fly' (e.g. by character-level operations or phoneme similarity). If a corrected version of a sentence is scored higher than the original by a LM, the proposed correction is assumed to be valid.

Finally, LMs and n-gram frequency can be used for ranking correction hypotheses, which is especially useful to determine the best correction from a set of alternatives. A typical use case is evaluating corrections from different systems or re-ranking an SMT system's n-best list of translations (Boroş et al., 2014; Felice et al., 2014; Gamon, 2010; Prokofyev et al., 2014; Yuan et al., 2016).

In any case, LMs are rarely used in isolation but rather in combination with other methods. For example, it is common to use sentence probabilities or perplexities as features for ML classifiers.

When working with error-annotated corpora, LMs are normally trained on the corrected versions of the data, so that judgements of test sentences are in accord with

the style and grammar of the training set. However, the number of correct instances in such datasets is often insufficient to build a fairly representative model of language, so they are generally replaced or augmented with larger native corpora, such as the British National Corpus or the English Gigaword corpus (see Section 2.2.1). One particularly popular dataset is Google's Web 1T 5-gram corpus (Brants and Franz, 2006), an enormous collection of n-grams collected from the web that is deemed one of the most comprehensive and updated for language modelling purposes. The availability of this dataset in other languages (Brants and Franz, 2009) is also a valuable resource for developing GEC systems outside of English.

Given the dynamic nature of language, however, some systems have made use of the web as corpus, retrieving n-gram counts from commercial search engines (Elghafari et al., 2010; Fallman, 2002; Gamon and Leacock, 2010; Hermet et al., 2008; Tetreault and Chodorow, 2009; Yi et al., 2008). While this ensures that systems will keep pace with language change, it is not free of other problems. As noted by Kilgarriff (2007), commercial search engines do not PoS-tag or lemmatise, counts are in terms of pages (not instances) and results often vary between different systems, to name only a few.

2.1.2.3 Classifiers

Machine learning classifiers are one of the most popular approaches to GEC. The main reason for this is that the most frequent error types, such as articles and prepositions, have limited confusion sets so their correction can be cast as a classification problem. A classifier receives a number of features representing the context of the analysed word or phrase in a sentence and outputs a predicted class that constitutes a correction. If the prediction is the same as the original item, the text is considered correct and left unchanged; otherwise, an error is flagged and the original word is replaced by the predicted correction. For example, a classifier for article correction would analyse each noun phrase in the text and predict the best class (i.e. article) from four possible alternatives: no article, *a*, *an* or *the*. The output classes will depend on the adopted strategy, so it is perfectly possible to use different reformulations to achieve the same result. In our article correction example, we could use only three classes (no article, indefinite, definite) and still encode the same information.

In order to build a classifier for error correction, the task must be framed as a decision problem with a finite number of alternatives (i.e. classes). Some error types, especially those involving closed-class words, are naturally well-suited for this approach, since they only allow a fixed set of possible corrections, such as articles, determiners, prepositions, subject-verb agreement, verb tenses, etc. Errors involving open-class words (such as word choice errors) must be restricted to closed confusion sets, which are generally compiled automatically from sample instances. Wu et al. (2010) for example, trained a classifier to suggest the most appropriate verb in

verb-object combinations, limiting predictions to 790 verbs. However, given the 'open' nature of content-word errors, classification is rarely used and other techniques, such as those based on compositional distributional semantics, are preferred instead.

Classifiers can certainly be useful for general error detection (Izumi et al., 2004, 2003) but their limitations make them unsuitable for general error correction. Nevertheless, they have been successfully used to correct specific error types, such as:

- articles (Berend et al., 2013; Bosch and Berck, 2013; Dahlmeier and Ng, 2011; Dahlmeier et al., 2012, 2011; De Felice and Pulman, 2008; Gamon, 2010; Han et al., 2006; Jia et al., 2013; Kunchukuttan et al., 2014, 2013; Lee, 2004; Minnen et al., 2000; Putra and Szabo, 2013; Quan et al., 2012; Rozovskaya et al., 2013; Rozovskaya and Roth, 2010c; Rozovskaya et al., 2012; Sakaguchi et al., 2012; Xiang et al., 2013; Yi et al., 2013; Zhang and Wang, 2014),

- prepositions (Bosch and Berck, 2013; Cahill et al., 2013b; Dahlmeier and Ng, 2011; Dahlmeier et al., 2012, 2011; De Felice and Pulman, 2008; Gamon, 2010; Jia et al., 2013; Quan et al., 2012; Rozovskaya et al., 2013; Rozovskaya and Roth, 2010b, 2011; Rozovskaya et al., 2012; Sakaguchi et al., 2012; Tetreault et al., 2010; Xiang et al., 2013; Yi et al., 2013; Zhang and Wang, 2014),

- noun number (Berend et al., 2013; Bosch and Berck, 2013; Jia et al., 2013; Kunchukuttan et al., 2014, 2013; Rozovskaya et al., 2013; Xiang et al., 2013; Yi et al., 2013; Yoshimoto et al., 2013),

- subject-verb agreement and verb forms (Bosch and Berck, 2013; Jia et al., 2013; Rozovskaya et al., 2013, 2014b),

- and others (Rozovskaya et al., 2014a, 2011; Wang et al., 2014a).

The most common classification techniques used in the literature include maximum entropy models, naive Bayes and support vector machines, to name just a few.

As is common in ML, finding good discriminative features is a key issue, especially when different error types are involved. Nevertheless, for the vast majority of syntactically-motivated errors, features such as contextual word and PoS n-grams, lemmas, phrase constituency information and dependency relations are generally useful (Felice and Yuan, 2014b; Leacock et al., 2014; Rozovskaya et al., 2013, p. 104).

Finally, another important aspect of classifier design concerns a more conceptual question: whether the correction problem should be modelled as a *selection* task. In that case, a classifier is trained to predict the most appropriate word given the context, ignoring the original one chosen by the writer and not using it as a feature. The task thus resembles a 'fill in the blank' exercise, where corrections only occur if the predicted word does not match the original. By contrast, modelling the problem as a correction task using the source word as a feature can enlighten the model about common confusions, leading to better performance (Rozovskaya and Roth, 2010c).

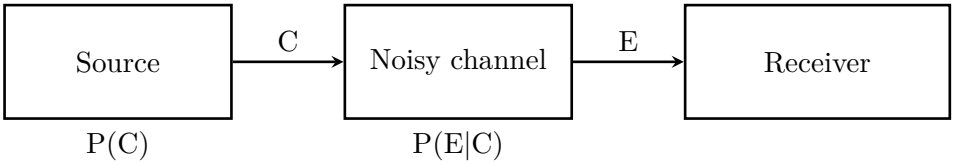


Figure 2.1: The noisy channel model.

In this case, care must be taken to balance the training set, since the high proportion of correct uses can make the classifier output the majority class (in this case, the source word), leaving the text uncorrected.

2.1.2.4 Machine translation

The correction of grammatical errors using machine translation (MT), and in particular SMT, has gained notable popularity in the last few years. Despite being originally developed for translation between different languages, SMT has been successfully applied to GEC, which is now seen as a translation problem from 'incorrect' into 'correct' English.

SMT is inspired by the *noisy channel* model (Shannon, 1948), which is mathematically formulated in Equation 2.1:

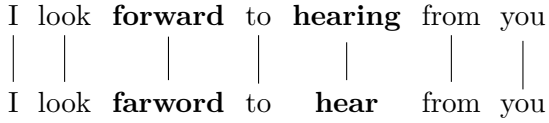
$$\hat{C} = \arg\max_C P(C|E) = \arg\max_C \frac{P(E|C)P(C)}{P(E)} = \arg\max_C P(E|C)P(C) \qquad (2.1)$$

In this model, a correct English sentence C is said to be corrupted by a noisy channel, resulting in an erroneous sentence E (see Figure 2.1). The goal of the model is to recover the correct intended sentence $\hat{C}$ using a LM of correct sentences $P(C)$ and a model of distortion in the channel $P(E|C)$ (a.k.a. the translation model). Candidate sentences are generated by means of a decoder, which normally uses a beam search strategy. As shown in Equation 2.1, the denominator $P(E)$ is ignored during computation since it is a constant for all correct sentences.

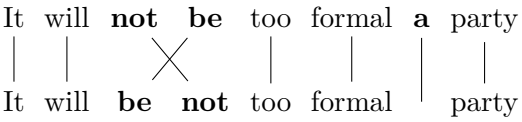
Unlike classifiers, which can be trained on native error-free corpora, SMT systems require a parallel corpus of translated (or corrected) instances for training, aligned at the sentence level. A LM is built on the set of corrected sentences although given the limited size of error-annotated corpora, bigger collections of native text can be used instead. The most common type of LM is an n-gram Markov model (Equation 2.2), where the probability of a sentence $(w_1, \ldots, w_m)$ is computed from the conditional probabilities of its words given their history (typically 2 to 5 words).

$$P(w_1, \ldots, w_m) = \prod_{i=1}^{m} P(w_i \mid w_1, \ldots, w_{i-1}) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, \ldots, w_{i-1}) \qquad (2.2)$$

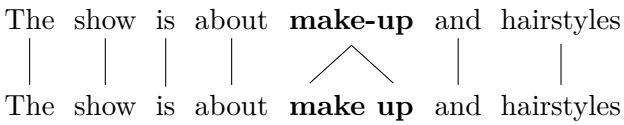
A translation model is built using probabilistic alignments of words and phrases between the source and target sentences, known as the IBM Models (Brown et al., 1993). The first of these models, IBM Model 1, estimates lexical translation probabilities using the Expectation Maximisation algorithm, e.g.:



IBM Model 2 adds an absolute word reordering model, so that translation probabilities are also dependent on the position of the input and output words, e.g.:

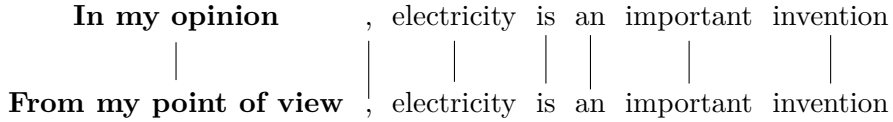


IBM Model 3 introduces the notion of *fertility*, which allows input words to generate more than one word in the output, e.g.:



IBM Model 4 improves on Model 3 by introducing a relative distortion model, so that the position of the translation of a word is influenced by the position of the translation of its preceding word in the input text. Finally, IBM Model 5 fixes a *deficiency* problem in Model 3 and 4 which allowed translations of different words to occupy the same position in the output text instead of filling vacant spaces. An alternative to IBM Models is the Hidden Markov Model (HMM) for word alignment which uses relative distortion but not fertility (Vogel et al., 1996).

These word-based models only allow one-to-many mappings and so are unable to represent many-to-one or many-to-many translations that are common in real life. This can be overcome by using phrase-based models, which use phrases (i.e. any arbitrary sequence of words) as translation units, e.g.:



In phrase-based models, the translation model $P(E|C)$ is further decomposed into:

$$P(\bar{E}_j^I \mid \bar{C}_j^I) = \prod_{i=1}^{m} \phi P(\bar{E}_i \mid \bar{C}_i) d(start_i - end_{i-1} - 1) \tag{2.3}$$

where $\phi P(\bar{E}_i \mid \bar{C}_i)$ represents the translation probability of phrase $\bar{E}_i$ given $\bar{C}_i$ (estimated from previous word alignments) and $d(start_i - end_{i-1} - 1)$ is a distance-based reordering model ($start_i$ and end_{i-1} refer to the positions of the first and last word of the source phrase that translates into the ith and $(i-1)$th phrase in the target). Phrase-based SMT models are trained on surface forms, so if test sentences contain words that have not been seen during training (a.k.a. out-of-vocabulary words), such words will remain untranslated (or uncorrected, in our case). This can be mitigated by the use of 'factored' models, which allow the integration of additional linguistic information at the word level, such as morphology and syntax. As part of our experiments in Chapter 4, we foray into PoS-factored models, where we use a PoS-tagged version of our datasets in an attempt to improve the accuracy of translations (i.e. corrections). During training, the PoS-tagged version of the data is used to build tables associating word forms to PoS tags (e.g. $p(\text{NN1} \mid \textit{house})$), generate PoS versions of the lexical phrases (e.g. *always will be* → *will always be* becomes RR VM VB0 → VM RR VB0) and build a PoS LM. The lexical and 'factored' components are then efficiently combined during decoding, using the same translation strategy as in ordinary phrase-based translation. For a more detailed description of these and other SMT models, please refer to Koehn (2010, p. 314).

The benefits of using SMT for GEC are manifold. First, SMT does not require expert knowledge but a parallel corpus of corrected text. This makes system development much more straightforward than with other ML approaches that require meticulous feature engineering. Second, there are no restrictions on error types (such as closed-class or open-class errors) so it is naturally well-suited to general error correction. Finally, SMT systems can correct nested and interacting errors simultaneously, avoiding pipelines of classifiers and complex combinations of individual results.

The use of SMT for GEC was pioneered by Brockett et al. (2006), who built a system to correct errors involving 14 countable and uncountable nouns which are often confusing for ESOL learners. Their training data comprised a large corpus of sentences extracted from news articles which were deliberately modified to include typical countability errors among Chinese learners. The resulting SMT system was generally able to beat the standard Microsoft Word 2003 grammar checker, although it produced a relatively higher rate of erroneous corrections.

Mizumoto et al. (2011) trained an SMT system for Japanese, using authentic learner sentences corrected by native speakers on a social learning network website. However, since the corpus was not annotated for error types, the resulting system was not restricted to particular error types. Their results validated that SMT can achieve high performance provided a sufficiently large corpus is used for training. These claims were later supported by Mizumoto et al. (2012), who carried out similar experiments on English text produced by Japanese students, and Behera and Bhattacharyya (2013), who applied hierarchical phrase-based SMT to correct errors in a corpus of essays written by non-native university students.

Ehsan and Faili (2013) trained SMT systems for correcting grammatical errors and context-sensitive spelling mistakes in English and Farsi, using artificial erroneous sentences. A comparison between the proposed systems and rule-based grammar checkers showed they are complementary, with a hybrid system achieving the best performance.

SMT has also been a popular approach among participants in the latest GEC shared tasks (see Section 2.3). Yoshimoto et al. (2013) built a translation system to correct article and preposition errors based on the work by Mizumoto et al. (2012), using the official training set plus additional corrected sentences from Lang-8 (see Section 2.2.2). Our own submission (Yuan and Felice, 2013) employed a PoS-factored SMT model to correct article, noun number, preposition, subject-verb agreement and verb form errors. Training data included the official shared task corpus (Section 2.2.2.2), parts of the CLC (Section 2.2.2.1) and artificially generated errors.

The top performing systems in the CoNLL 2014 shared task (see Section 2.3.2) have demonstrated that SMT can yield state-of-the-art performance on general error correction. Our winning system (Felice et al., 2014) is a pipeline of a rule-based system and an untuned lexical phrase-based SMT system, which produces candidate corrections. These corrections are then used to generate full corrected sentences which are ranked by a web-based LM and finally filtered by error type. Our work showed that an untuned SMT system can achieve state-of-the-art performance on the task, while hypothesis re-ranking improves SMT output and type filtering can improve final results. The third best system (Junczys-Dowmunt and Grundkiewicz, 2014) used a phrase-based SMT model that included specialised features and was optimised for F-score, concluding that parameter optimisation is essential. Kunchukuttan et al. (2014) also explored tuning on $F_{0.5}$ to increase precision, but results showed it worsened performance. Other participating teams using SMT models include Wang et al. (2014b), who trained factored models encoding word stem, prefix, suffix and PoS information; and Wu et al. (2014), who used a basic phrase-based model to correct interacting errors.

SMT-based GEC systems have also been proposed for Chinese (Zhao et al., 2014, 2015) and Arabic (Bouamor et al., 2015; Bougares and Bouamor, 2015; Jeblee et al., 2014; Mostefa et al., 2015, 2014; Tomeh et al., 2014).

Other approaches using MT for error correction have not aimed to train SMT systems but rather use them as auxiliary tools. One such example is the use of *round-trip* translations (i.e. translations into a pivot foreign language and back into English) for correcting preposition errors in learner writing (Hermet and Désilets, 2009; Madnani et al., 2012). Conversely, the noisy channel approach has been used for GEC without framing the task as an MT problem. These approaches make use of a beam search decoder to produce a sentence-level correction from the output of other components (Buys and Merwe, 2013; Dahlmeier and Ng, 2012a; Park and Levy, 2011; West et al., 2011; Wilcox-O'Hearn, 2013).

Despite their success in GEC, SMT-based systems suffer from a few shortcomings. For example, they tend to produce locally well-formed phrases with poor overall grammar, they exhibit a predilection for changing phrases to more frequent versions (even when the original is correct), they are unable to process long-range dependencies and they are hard to constrain to particular error types. Last but not least, the performance of SMT systems depends largely on the amount of parallel data used for training, which is very limited for GEC. A common solution to this problem is to generate artificial datasets, where errors are injected into well-formed text to produce pseudo-incorrect sentences, as described in Section 2.2.3.

2.1.2.5 Other approaches

Apart from the most common approaches, there have been efforts to tackle error detection and correction using other techniques, including: integer linear programming to enforce grammatical compatibility between corrections (Rozovskaya and Roth, 2013; Wu and Ng, 2013), compositional distributional semantics to correct content word combinations (Kochmar and Briscoe, 2013) and, more recently, deep neural networks for different error types (Sun et al., 2015; Xie et al., 2016).

2.2 Data

Modern GEC systems are generally built using data-driven approaches, where knowledge is extracted from sample texts rather than explicitly coded by experts (see Section 2.1.2). The type of corpora used for training can vary, depending on factors such as the purpose of the system, the error types addressed and the availability of datasets. The following sections outline the three main types of data that are often used in the field, with a detailed description of the corpora used in our work.

2.2.1 Native data

Before the first error-annotated datasets became available, GEC systems were trained on native corpora, typically news text. Well-formed sentences can only be used to learn models of correct usage, so errors are flagged only when there are discrepancies between the models' predictions and the evaluated text. Error-free corpora are also essential to build LMs.

Some of the most popular datasets used in GEC are collections of newspaper text, such as English Gigaword (Parker et al., 2011) or the Wall Street Journal corpus (Fallman, 2002). Unfortunately, news articles do not resemble real learner text so many types of constructions that are common among learners will not be properly represented in these corpora, such as the use of the first person, questions, salutations and valedictions, etc. The British National Corpus (BNC)[4], on the other

[4] http://www.natcorp.ox.ac.uk/

hand, is a more 'balanced' dataset. This 100-million-word corpus contains samples of written and spoken British English from the late 20th century, including extracts from newspapers, books, academic literature, letters, and school and university essays, among other genres.

The web has also been used as a source of normative text. Wikipedia is probably the most used resource, including its 'simple English' version to a lesser extent. This latter dataset might in fact be more appropriate to model learner writing, given the lower proportion of complex structures and technical terms. Some more innovative uses of Wikipedia for GEC are based on the extraction of corrections from article revisions (Cahill et al., 2013b; Grundkiewicz and Junczys-Dowmunt, 2014). However, care must be taken when using these edits, as many of them are purely stylistic or content-related rather than grammatical. Two other popular resources are ukWaC (Baroni et al., 2009), a 2-billion-word collection of web pages from the .uk domain, and the Google Web 1T 5-gram corpus (Brants and Franz, 2006), containing over 3.7 billion n-grams and their counts collected from the web.

2.2.2 Learner data

The growing popularity of corpus linguistics as a way of studying language from empirical evidence has paved the way for the creation of many specialised corpora. Granger (2003a) defines 'learner corpora' (also called 'inter-language' or 'L2 corpora') as 'electronic collections of authentic foreign or second language data'. These collections have become very valuable resources for studying language development and assist the creation of educational resources, from printed learning material to software applications (Antoniadis et al., 2006; Gillard and Gadsby, 1998; Granger, 2003a; Tono, 2003).

The usefulness of learner corpora depends largely on their annotation. This can be done at various levels and typically covers background learner information (e.g. age, sex, nationality, L2 proficiency), script-level information (e.g. genre, prompt, score) and detailed error annotation. The latter involves the adoption of guidelines for marking and correcting errors, which are often corpus-dependent, as well as a suitable error typology, if errors are to be classified into categories. Error annotation can be a painstaking process so most available datasets are generally not annotated for errors.

Given its status as the world's most learnt L2, it is no surprise that most learner corpora are for English. The Longman Learners' Corpus (LLC)[5] is one of the earliest examples, which amounts to c. 10 million annotated essays and exam scripts written by students of different nationalities. The information in the LLC is a valuable resource for in-house lexicographers and textbook authors tailoring their material to the students' needs (Gillard and Gadsby, 1998).

[5] http://www.pearsonlongman.com/dictionaries/corpus/learners.html

The International Corpus of Learner English (ICLE) (Granger, 2003b) is another popular dataset comprising argumentative essays written by upper-intermediate and advanced students with a wide range of L1s. The current version of the corpus contains around 3 million words but lacks error annotations.

Two of the largest error-annotated learner corpora have emerged from collaborative projects at the University of Cambridge: the CLC (described in Section 2.2.2.1) and the EF-Cambridge Open Language Database (EFCamDat) (Geertzen et al., 2012). EFCamDat contains over 70 million words in 1.2 million essays written by nearly 175 learners with a variety of L1 backgrounds. These texts correspond to assignments submitted to EnglishTown (now EF English Live[6]), an online courseware platform by EF Education First, and span across all levels of the Common European Framework of Reference for Languages (CEFR)[7].

In recent years, a series of corpora have been automatically compiled from Lang-8,[8] a language exchange website where native speakers correct short compositions submitted by learners from around the world. These corpora[9] are available for many languages and have found favour with the NLP community (Junczys-Dowmunt and Grundkiewicz, 2014; Mizumoto et al., 2012; Sakaguchi et al., 2013; Sawai et al., 2013; Yoshimoto et al., 2013). A thorough list of 'Learner Corpora Around the World' is maintained online by the Centre for English Corpus Linguistics at the Université Catholique de Louvain.[10] Detailed surveys of available learner English corpora are provided by Pravec (2002) and Schiftner (2008). In what follows, we describe two major error-corrected learner corpora that are used for our experiments.

2.2.2.1 Cambridge Learner Corpus

The CLC is a proprietary collection of non-native English texts compiled by Cambridge University Press and Cambridge English Language Assessment (Nicholls, 2003). Since its inception in 1993, the corpus has grown to include 52.2 million words across 200,000 exam scripts written by students from 148 L1 backgrounds, and continues to expand. Each of the scripts in the collection has been written in response to an examination prompt and includes metadata such as the student's age and L1, prompt indexes and scores. Genres represented in the data include letters, articles, reports and short stories, among others.

All these scripts come from ESOL examinations in the A2–C2 range of CEFR levels and have been transcribed verbatim, retaining all original errors. A portion of the corpus (about 25.5 million words) has been manually error-coded by two experts

[6] `http://englishlive.ef.com`

[7] The CEFR provides a framework for the assessment of language proficiency, defined by six levels of attainment: A1–Breakthrough (beginner), A2–Waystage (elementary), B1–Threshold (intermediate), B2–Vantage (upper intermediate), C1–Effective Operational Proficiency (advanced) and C2–Mastery (proficient). Details can be found in its original specification (Council of Europe, 2001).

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

[9] `http://cl.naist.jp/nldata/lang-8/`

[10] `http://www.uclouvain.be/en-cecl-lcworld.html`

```
Since the <NS type="RP"><i>internet</i><c>Internet</c></NS> was
introduced, many of us <NS type="TV"><i><NS type="RV"><i>wouldn't
have imagined</i><c>couldn't have imagined</c></NS></i><c>can't
imagine</c></NS> <NS type="UD"><i>the</i></NS> <NS type="FN"><i><NS
type="IN"><i>lifes</i><c>lives</c></NS></i><c>life</c></NS> without
<NS type="RA"><i>this</i><c>it</c></NS>.
```

Figure 2.2: A sample annotated sentence from a public portion of the CLC. The original version **Since the internet was introduced, many of us wouldn't have imagined the lifes without this.* is corrected to *Since the Internet was introduced, many of us can't imagine life without it.*

according to a taxonomy of 80 error types specifically designed for the CLC (see Appendix A). The adopted annotation scheme uses the following notation, which permits nested errors:

```
<NS type=CODE><i>error</i><c>correction</c></NS>
```

Each error code is composed of two letters. The first of them indicates the general type of error from among F (wrong word form), M (something missing), R (replacement), U (unnecessary word or phrase) and D (wrong word derivation). Types M, R and U can occur in isolation. The second letter in the code indicates the word class of the required correction, including A (pronoun), C (conjunction), D (determiner), J (adjective), N (noun), Q (quantifier), T (preposition), V (verb) and Y (adverb). Other special codes are introduced for errors beyond these categories, such as 'AG' for agreement and 'S' for spelling errors.

This coding system allows researchers to look at different cross sections of the data by specifying only one of the two letters in the error code. For example, '*T' can expose all preposition errors, regardless of their cause. Figure 2.2 shows a sample annotated sentence from the CLC.

A subset of essay scripts from the CLC corresponding to the First Certificate in English (FCE), a main-suite examination at the B2 level, was made publicly available[11] for non-commercial purposes in 2011 (Yannakoudakis et al., 2011).

The dataset comprises 1,244 anonymised scripts from 2000 and 2001 written by learners with sixteen different L1s. Each of the scripts contains answers to two writing tasks between 120 to 180 words and includes the age group and L1 of the student along with an overall mark given by the examiners. The prompts eliciting these assignments are also provided.

The FCE data has been used for several purposes, including GEC (Dale et al., 2012; Felice et al., 2014; Rozovskaya et al., 2014b; Seo et al., 2012; Yuan and Felice, 2013), native language identification (Brooke and Hirst, 2012a,b; Bykh and Meurers, 2014; Bykh et al., 2013; Kochmar, 2011) and automated essay scoring (Yannakoudakis et al., 2011).

[11] http://ilexir.co.uk/applications/clc-fce-dataset/

```
S There is hundred of ways that an idea can originate from .
A 1 2|||SVA|||are|||REQUIRED|||-NONE-|||0
A 2 3|||Wform|||hundreds|||REQUIRED|||-NONE-|||0
A 5 6|||Wci|||from which|||REQUIRED|||-NONE-|||0
A 10 11|||Prep||||||REQUIRED|||-NONE-|||0
```

Figure 2.3: A sample annotated sentence from NUCLE. The prefix S specifies the original sentence while the prefix A indicates annotations. Each annotation contains the start and end token offset of the error (zero-based), a given type, a correction, an indication of whether the correction is strictly required, an extra field for supplementary data and the annotator's ID at the end. In this example, the resulting corrected sentence is *There are hundreds of ways from which an idea can originate.*

2.2.2.2 NUCLE

The National University of Singapore Corpus of Learner English (NUCLE) comprises 1,414 essays written in English by non-native undergraduate students at the National University of Singapore (Dahlmeier et al., 2013). The essays were written in response to prompts about a wide range of topics (such as healthcare, technology and environmental pollution) and are over 500 words in length on average. The corpus includes error annotations provided by 10 native English instructors (although none of the sentences were multiply annotated), based on an initial taxonomy of 27 error types. The original corpus is a collection of SGML files annotated at the character level although an equivalent but simpler tokenised plain-text format is generally used instead for compatibility with the M^2 Scorer (see Section 3.1.3). An example annotated sentence using this simplified format is shown in Figure 2.3.

The NUCLE corpus is freely available for research purposes and was used as the official training data in the CoNLL 2013 and 2014 GEC shared tasks (see Section 2.3.2). The latest version, NUCLE 3.2, classifies errors into 28 types, after the addition of a separate category for preposition errors (Ng et al., 2013). The current error typology is shown in Appendix B.

2.2.3 Artificial data

The scarcity of error-annotated data in the early years of GEC has made researchers consider alternative ways of training their systems. As noted in Section 2.1, many have resorted to native well-formed text that is used to build models of correct language; however, erroneous test data is still needed for evaluation.

A simple strategy to overcome this problem is the deliberate injection of errors in well-formed sentences as a replacement for genuine erroneous text. In fact, evaluating on artificial test sets is quite common in error detection and correction tasks, such as spelling correction (Bigert, 2004; Fossati and Di Eugenio, 2008; Wilcox-O'Hearn et al., 2008; Wilcox-O'Hearn, 2014), sentence grammaticality judgements (Islam and Inkpen, 2011; Wagner et al., 2007, 2009) and pronunciation error detection (Herron et al., 1999; Kanters et al., 2009; Zhao et al., 2012). As noted by Dickinson (2010),

using artificial datasets for evaluation has at least one advantage over real learner text, in that the errors and their corrections are known in advance.

Using artificial errors as training data is a more complex matter. Data-driven approaches to GEC typically require large collections of positive and negative instances for training so the effects of artificial generation become more evident. In order to learn useful patterns, systems should be given errors that are plausible in specific contexts and resemble actual learner errors. As explained in Section 1.2, errors committed by non-native speakers are not random but largely influenced by their L1s. Thus, any replication of learner errors should be based on a prior analysis of real data in order to mimic them realistically.

Artificial error generation (AEG) allows researchers to create very large error-annotated corpora with little effort and control variables such as topic and error types. A number of AEG methods have been proposed for GEC, including deterministic and probabilistic approaches, which we review below.

2.2.3.1 Deterministic approaches

In this category, we include approaches that generate errors in systematic ways which do not make use of learner error distributions. These methods invariably create an error for every relevant instance in a given corpus (e.g. an article or preposition) and replace them with another from a predefined set of alternatives. In some cases, the selection of a replacement is random, but in others it is still predefined and not based on empirical error probabilities.

In their pioneering work, Izumi et al. (2003) described a system trained on artificial data to correct article errors made by Japanese learners of English. A corpus was created by replacing *a*, *an*, *the* or the zero article with a different option chosen at random in more than 7,500 correct sentences, which was then used to train a maximum entropy model. Results showed an improved detection rate for omission errors but no change for replacement errors when compared to using just the original corpus. In addition, precision rose significantly while recall remained the same.

Sjöbergh and Knutsson (2005) created an artificial corpus of split compounds and word order errors, two of the most frequent types among non-native Swedish speakers. A system trained on these synthetic instances was then used to detect errors in real learner data, outperforming state-of-the-art grammar checkers. Unlike Izumi et al. (2003), the authors found that artificial errors increased recall but lowered precision.

Brockett et al. (2006) described the use of an SMT system for correcting a set of 14 countable/uncountable nouns which are often confused by learners of English. A training corpus was built using sentences from news articles that were deliberately modified to include typical countability errors observed in a Chinese learner corpus. However, their approach was still deterministic, since they used hand-coded rules to change quantifiers (e.g. *much* → *many*), generate plurals (e.g. *advice* → *advices*) and insert unnecessary determiners.

Lee and Seneff (2008) created an artificial corpus of verb form errors by changing verbs in error-free text to a different form (bare infinitive, *to*-infinitive, third person singular present, past, *-ing* participle and *-ed* participle). The authors then investigated how these errors affected parse trees and used this information to improve the correction of subject-verb agreement, auxiliary agreement and complementation errors.

Ehsan and Faili (2013) used SMT with AEG to correct grammatical errors and context-sensitive spelling mistakes in English and Farsi. Training corpora were obtained by injecting artificial errors into well-formed treebank sentences using predefined error templates. Whenever an original sentence from the corpus matched one of the templates, a pair of correct and incorrect sentences was generated. The process was repeated multiple times if a sentence matched more than one error template, thereby generating many pairs for the same original sentence. A comparison between the proposed systems and rule-based grammar checkers showed that they are complementary, with a hybrid system achieving the best performance.

2.2.3.2 Probabilistic approaches

A few researchers have explored probabilistic methods in an attempt to mimic real data more accurately. Wagner et al. (2007, 2009), for example, created a corpus of artificial errors by distorting sentences from the BNC. Errors were generated probabilistically for the four most frequent error types observed in a corpus of ungrammatical sentences. Experiments with a number of methods (including PoS n-grams and decision trees) showed that artificial data is effective in detecting the targeted errors, with only poorer performance on missing words. Accuracy was also found to drop when testing on real learner texts as opposed to synthetic test data, confirming that training and test data should be as similar as possible.

A general approach to AEG is presented by Foster and Andersen (2009), who describe a tool (GenERRate) for generating errors based on given patterns. The tool operates on an input corpus of well-formed sentences and an error analysis file, which can be hand-coded or created automatically from an error-annotated corpus. This analysis file contains a list of operations to generate errors, such as deleting an article, changing the tense of a verb or moving words within the sentence. If users specify a proportion for each operation, the tool will attempt to create errors until the required proportion is achieved or all the sentences in the input corpus are exhausted. When no frequency information is specified, the tool behaves deterministically, trying to inject one error per type in each of the available sentences.

A replication of the experiments in Wagner et al. (2009) using GenERRate showed an improvement in accuracy, suggesting that the proposed method can produce artificial errors more accurately. In a second set of experiments, the authors created a new corpus mimicking errors in the CLC, which was then used to train a classifier to detect grammatical sentences. Evaluation on a held-out portion of the CLC showed a decrease in accuracy when compared to training on real errors; however, much of

this loss was recovered when both types of data were used, suggesting that artificial errors could be used to augment real learner data. Still, performance was better when using learner data on its own, so it is likely that the recovered performance is actually due to genuine data. Yet, the generic operations defined in GenERRate make it possible to generate errors of any type, unlike previous methods.

The formulations by Rozovskaya and Roth (2010c) constitute perhaps the most sophisticated work on probabilistic AEG, including elaborate methods for creating article (Rozovskaya and Roth, 2010c) and preposition errors (Rozovskaya and Roth, 2010b, 2011) based on statistics from an ESOL corpus. Errors were injected into Wikipedia sentences following different strategies:

General Target words (e.g. articles) were replaced with others of the same class with probability x (varying from 0.05 to 0.18). Each new word was chosen uniformly at random.

Distribution before correction (in ESOL data) Target words in the error-free text were changed to match the distribution in a learner corpus before annotation is performed.

Distribution after correction (in ESOL data) Target words in the error-free text were changed to match the distribution in a learner corpus after annotation is performed.

L1-specific error distribution Errors were injected according to observed confusions for the given L1. More specifically, if we estimate P(source|target) from an error-annotated corpus (i.e. the probability of an incorrect *source* word being used when the correct *target* is expected), we can generate more accurate confusion sets where each candidate has an associated probability depending on the observed word. For example, if a group of learners use the preposition *to* in 10% of cases where the preposition *for* should be used (i.e. P(source=*to*|target=*for*)= 0.10), we can replicate this error by replacing the occurrences of the preposition *for* with *to* with a probability of 0.10. When the source and target words are the same, P(source=x|target=x) expresses the probability that a learner produces the expected word.

Experiments using the aforementioned methods produced better results than using uniform distributions where all errors and corrections are equally probable. In particular, classifiers trained on artificially generated data outperformed those trained on native error-free text (Rozovskaya and Roth, 2010c, 2011). However, previous work has shown that using artificially generated data as a replacement for non-native error-corrected data can lead to poorer performance (Foster and Andersen, 2009; Sjöbergh and Knutsson, 2005). This would suggest that artificial errors are more useful than native data but less useful than annotated non-native data.

Rozovskaya and Roth (2010c) also controlled other variables in their experiments. On the one hand, they only evaluated their systems on sentences that had no spelling mistakes so as to avoid degrading performance. This is particularly important when training classifiers on features extracted with linguistic tools (such as parsers or taggers) as they could provide inaccurate results for malformed input. On the other hand, the authors worked on a limited set of error types (mainly articles and prepositions) which are closed word classes and therefore have reduced confusion sets. Thus, it would be interesting to investigate how their ideas extrapolate to open-class error types, like verb form or content word errors.

Because errors are generally sparse (and therefore error rates are low), replicating errors based on observed probabilities can easily lead to low recall. In order to address this issue during AEG, Rozovskaya et al. (2012) proposed an *inflation method* that boosts confusion probabilities in order to generate a larger proportion of artificial instances. This method has been shown to improve F-scores when correcting determiners and prepositions and has since been adopted by other researchers (Felice and Yuan, 2014a; Putra and Szabo, 2013; Rozovskaya et al., 2013, 2014a).

Dickinson (2010) presented a probabilistic approach to generate artificial syntactic and morphological errors for Russian. Errors were created using different random methods subject to constraints derived from a prior error analysis (such as PoS properties). The resulting dataset was evaluated on PoS-tagging accuracy in an attempt to study the robustness of taggers on malformed input.

Imamura et al. (2012) replicated the probabilistic method by Rozovskaya and Roth (2010c) in order to generate artificial incorrect sentences for Japanese particle correction, with inflation factors ranging from 0.0 (no errors) to 2.0 (double error rates). Their results showed that the performance of artificial corpora depends largely on the inflation rate but it can be improved via domain adaptation.

In a more exhaustive study, Cahill et al. (2013b) investigated the usefulness of automatically-compiled sentences from Wikipedia revisions for correcting preposition errors. A number of classifiers were trained using error-free text, automatically-compiled annotated corpora and artificial sentences based on error probabilities from Wikipedia revisions and Lang-8. Their results revealed that artificial errors provide competitive results and perform robustly across different test sets. A learning curve analysis also showed that system performance increases as more training data is used, both real and artificial.

More recently, some teams have also reported improvements by using artificial data in their submissions to the CoNLL 2013 & 2014 shared tasks (see Section 2.3.2). Rozovskaya et al. (2013, 2014a) applied their inflation method to train a classifier for determiner errors that achieves state-of-the-art performance while Yuan and Felice (2013) and Felice et al. (2014) used naively-generated artificial errors within an SMT framework to increase recall.

2.3 Shared tasks

Many of the recent advances in GEC have emerged from a series of competitions or *shared tasks* that have encouraged researchers to develop error correction systems. These efforts have also helped to provide an organised framework for research in the field, which includes publicly available test data and the use of specific evaluation metrics. Below we review GEC shared tasks for English and touch on similar tasks for other languages and purposes.

2.3.1 Helping Our Own 2011 & 2012

The Helping Our Own (HOO) 2011 shared task was designed to promote the development of automatic tools that could assist non-native speakers of English in writing scientific papers in the field of NLP (Dale and Kilgarriff, 2011). Participants were given a training set consisting of 19 text fragments taken from papers in the ACL Anthology (Bird et al., 2008). Fragments were an average of 940 words in length and contained a total of 1,264 edits to correct errors and infelicities, provided by two professional copy-editors (Dale and Narroway, 2011). A similar set containing 1,057 edits was given for testing. Errors were annotated following the CLC error annotation scheme, with most of them involving articles, prepositions, punctuation and wrong word choice.

The participating teams were allowed to build their systems using additional resources and submit up to 10 different 'runs', so that they could provide alternative system outputs.

Evaluation was performed in terms of precision, recall and F-score (see Section 3.1.1) at three different levels: detection (the system determined that an edit was necessary), recognition (the system correctly identified the extent of an edit) and correction (the system provided one of the corrections in the gold standard). 'Bonus' scores were given to systems that matched optional edits. For detailed information, refer to Section 3.1.2. The six systems that took part in the task used different approaches, with the best system using a pipeline of classifiers (Rozovskaya et al., 2011). The maximum F_1-scores obtained for detection, recognition and correction were 36.6, 33.7 and 30.6 respectively, reflecting the difficulty of the task.

The second edition of the shared task, HOO 2012, focused only on article and preposition errors (Dale et al., 2012). This time, the training data was a subset of the public FCE corpus, containing 1,000 exam scripts for training and 100 for testing. Evaluation was performed at the same three levels as in the previous year, with the exception that recognition also involved the correct identification of error types.

The fourteen participating teams were again allowed to submit up to 10 'runs' plus additional annotations for errors that were rightly corrected by their systems but not included in the gold standard. System performance varied widely across error types and evaluation levels. The best two systems used a pipeline of classifiers.

On the one hand, the NUS system (Dahlmeier et al., 2012) achieved the best scores for correction, both before ($F_1 = 28.7$) and after revisions ($F_1 = 37.83$), when it also scored highest for recognition ($F_1 = 42.52$). The UI team (Rozovskaya et al., 2012), on the other hand, scored higher for detection ($F_1 = 40.2$) and recognition ($F_1 = 35.39$) before revisions but only highest for detection ($F_1 = 43.24$) afterwards.

2.3.2 CoNLL 2013 & 2014

These shared tasks were organised as part of the Conference on Natural Language Learning (CoNLL). The first edition in 2013 focused on five error types: articles and determiners, noun number, prepositions, subject-verb agreement and verb form errors (Ng et al., 2013). The training data provided for the task included 1,397 essays from the NUCLE corpus, which were written in English by students at the National University of Singapore (see Section 2.2.2.2). Essays were corrected by professional English instructors and only one official correction was given for each annotated error. A preprocessed version of the data containing only the 5 relevant error types, out of the total 27 in the original typology, was also provided. The test data comprised 50 new essays written in response to two prompts, one of which did not overlap with prompts in the training data.

Systems were allowed to use additional resources as long as they were publicly available. Evaluation was performed using the MaxMatch (or M^2) Scorer, which computes precision, recall and F_1 on automatically extracted phrase-level edits (see Section 3.1.3). As in HOO 2012, teams were given the chance to submit alternative corrections after the first evaluation round.

The task attracted submissions from 17 teams that employed different correction strategies. The top performing system (Rozovskaya et al., 2013) used a set of classifiers and achieved an overall F_1-score of 31.20 and 42.14 before and after alternative answers were considered, respectively.

The second edition of this shared task (Ng et al., 2014) extended correction to all 28 types in updated versions of the training and test data. The new test set contained corrections from two independent expert annotators. Based on the assumption that precision is more valuable than recall in GEC environments, system evaluation was changed to use $F_{0.5}$, weighting precision twice as much as recall.

A total of 13 teams submitted their systems' output. Our hybrid system (Felice et al., 2014), integrating rule-based and SMT components with LM ranking and type filtering, scored first ($F_{0.5} = 37.33$) and second ($F_{0.5} = 43.55$) before and after alternative corrections. The runner-up (Rozovskaya et al., 2014a), a combination of ML classifiers addressing only 9 error types, achieved the second ($F_{0.5} = 36.79$) and first place ($F_{0.5} = 45.57$) respectively. Results on this task revealed that SMT approaches could be highly competitive and achieve state-of-the-art performance, as two out of the three best performing systems used such models.

The test sets used in the CoNLL shared tasks remain freely available, serving as common datasets for future comparisons.

2.3.3 Other shared tasks

Similar shared tasks have also been organised for languages other than English. The NLP-TEA-1 (Yu et al., 2014) and NLP-TEA-2 (Lee et al., 2015) shared tasks focused on Chinese grammatical error diagnosis. The tasks concerned two aspects: 1) binary error detection, where a sentence is classified as correct or incorrect, and 2) error identification, which classifies an incorrect sentence as having only one error from the given typology (redundant word, missing word, word disorder and incorrect word selection).

The QALB-2014 (Mohit et al., 2014) and QALB-2015 (Rozovskaya et al., 2015) shared tasks focused on automatic text correction for Arabic, originally for native speakers but later extended to include learner text. The training and test data included errors from many categories (e.g. spelling, punctuation, word choice, morphology, syntax and usage) but systems were not required to classify them, only to correct them.

Other proposed shared tasks not directly focused on GEC can also be relevant for this purpose, such as the ones for Automated Evaluation of Scientific Writing (AESW) (Daudaravicius et al., 2016) and Automatic Post-Editing (APE) for MT (Bojar et al., 2015).

Chapter 3

Evaluation methods

This chapter gives an overview of evaluation methods for GEC. The first section describes early approaches based on traditional metrics, such as precision, recall and F-measure. In the second section, we propose the I-measure, a new evaluation method that overcomes the limitations of previous approaches. In the third section, we review recent work on the use of crowdsourcing as a means to produce human rankings and develop more accurate metrics. Finally, we summarise the status quo and discuss the problems inherent in evaluation metrics for GEC.

3.1 Previous work

3.1.1 Traditional evaluation metrics

GEC systems are generally evaluated by comparing their output (a hypothesis) with corrections in a gold standard (a reference). This comparison is often made in terms of the 'edits' that the system and gold standard propose to produce a corrected version of the source text. For example, for the sentence **I like swim*, some possible correction edits are (*swim* → *swimming*) and (*swim* → *to swim*). Edits can be expressed in many ways but they are most often phrases or tokens.

When using a gold standard, each system edit can be classified as a true positive (TP), a false negative (FN), a false positive (FP) or a true negative (TN); sometimes depending on whether detection or correction performance is being evaluated:

- A TP is a 'hit', a match between the system and the gold standard. For detection, flagging the error will suffice but for correction, the system's hypothesis must also match the gold standard correction exactly.

- A FN is a 'miss', an error annotated in the gold standard that the system failed to catch. For correction, this includes cases where an error is detected but the provided correction does not match the gold standard.

	Writer	**Annotator**	**System**
TN	X	X	X
FP	X	X	Y
FN	X	Y	X
TP	X	Y	Y
*****	X	Y	Z

Table 3.1: Classification under the WAS evaluation scheme (Chodorow et al., 2012).

Writer	She	love	skying	in	the	alps	
Annotator	She	loved	skiing	in	the	Alps	.
System	She	loves	skying	on	the	Alps	
Detection	TN	TP	FN	FP	TN	TP	FN
Correction	TN	FP+FN	FN	FP	TN	TP	FN

Table 3.2: An example of token-level WAS evaluation.

- A FP is a 'false alarm', a correction proposed by the system that is not in the gold standard.
- A TN is a correct piece of text that is duly preserved by the system. In other words, it is an item that is correctly identified by the system as not needing correction. Because most words in learner text are generally correct, TNs constitute the majority class. These edits are normally ignored and not included in the gold standard.

This classification, referred to as the Writer-Annotator-System (WAS)[1] evaluation scheme in GEC (Chodorow et al., 2012), is summarised in Table 3.1. An example is given in Table 3.2.

As noted by the authors, classification varies for detection and correction when all three values are different (the last row in Table 3.1). For detection, this case is always a TP while for correction it is both a FP for Z (because Writer $\neq$ System) and a FN for Y (because Annotator $\neq$ System). This will become more relevant when we introduce a new measure in Section 3.2.

As in other areas of NLP, these counts are used to compute the traditional metrics of precision (P), recall (R) and F-measure (F). Definitions of P and R are given in Equation 3.1 and 3.2 respectively.

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad (3.1) \qquad\qquad \mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad (3.2)$$

Equation 3.1 shows that P is the proportion of actual errors among the ones predicted by the system. It can thus be seen as a measure of 'purity' or 'quality',

[1]The authors use different terminology, where 'writer' refers to the source (original text), 'annotator' to the reference correction (gold standard) and 'system' to a system's correction hypothesis.

indicating how good or reliable detection (or correction) is in general. The lower the number of 'false alarms', the higher the precision. R gives the proportion of errors captured by the system out of all the errors in the text, and is therefore a measure of 'coverage'. As more errors are successfully detected (or corrected), recall will be higher. The trade-off between P and R is a well-known problem in information retrieval, as it is difficult in practice to achieve high scores in both metrics at the same time (Buckland and Gey, 1994).

Computing P and R for our example in Table 3.2 yields $P = 2/(2+1) = 2/3$ and $R = 2/(2+2) = 1/2$ for detection, and $P = 1/(1+2) = 1/3$ and $R = 1/(1+3) = 1/4$ for correction. As expected, scores are lower for correction, since it requires stricter matching.

P is typically considered more important than R in GEC for at least two reasons. First, finding a large number of errors is not as important as providing good corrections. This is crucial in a learning environment, where we would much rather catch fewer errors than mark correct usage as incorrect. As noted by Leacock et al. (2014, p. 11), a system should 'minimize false positives, which are notoriously annoying for users'. Second, P can be a useful indicator of the effect of the system on the source text. In particular, whenever $P > 0.5$, the error rate decreases (and therefore accuracy increases, see below), so the original text is improved. This is because, in theory, applying more correct edits than incorrect edits will yield a positive balance. However, in practice, this depends on the edits, especially if they are phrases of variable length.

In other scenarios, however, R might be preferred. For example, teachers using GEC systems as a tool for preliminary correction may prefer to check a large number of potential errors rather than leave some errors uncorrected. Likewise, users who need to proofread very important documents might also be willing to review a few false positives to ensure no error has slipped.

We can also compute F to get a unified measure of performance, as shown in Equation 3.3:

$$F_\beta = (1+\beta^2) \times \frac{P \times R}{(\beta^2 \times P) + R} \tag{3.3}$$

Parameter β is a positive real that controls the importance of R with regard to P. The most common definition (F_1, or simply F) equals the harmonic mean of P and R, where both values are considered equally important (Equation 3.4):

$$F_1 = 2 \times \frac{P \times R}{P + R} \tag{3.4}$$

For our example, this gives $F_1 = 4/7$ for detection and $F_1 = 2/7$ for correction.

Because P, R and F are built upon TPs, they can only give meaningful results if $TP \geq 1$, that is, if the system gets at least one correction right. Consequently, the difference between a 'do nothing' baseline and a system that only makes the text

worse is indiscernible. The axiomatic definition of P and R when their denominators are zero might also pose problems if the metrics are analysed in isolation (e.g. $P = 1$ might occur by definition). Finally, these metrics do not provide a clear indication of how the error rate varies after applying a system's corrections.[2]

These limitations can be overcome by using accuracy (Acc), which computes the proportion of correct cases over the total number of cases, as shown in Equation 3.5:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3.5}$$

However, Acc can easily be dominated by TNs in problems where the majority of cases belong to the negative class (Manning and Schütze, 1999, p. 269). This is also the case in GEC, where errors constitute a minority. For example, if article errors occur only 15% of the time in a given corpus, leaving the text uncorrected will achieve an accuracy of 85%. In our particular example, $\text{Acc} = 4/7$ for detection and $\text{Acc} = 3/7$ for correction.[3]

Acc can only be computed in cases where we can enumerate all TNs, which is why it has been mostly used for article and preposition errors (De Felice and Pulman, 2008; Rozovskaya and Roth, 2010c). Extending this approach to other error types involves the identification of all *relevant instances* or positions where an error can occur, which is not always easy and renders the evaluation process costly, language-dependent, and possibly inexact.[4] An easier way to circumvent this problem is to define a clear-cut unit of evaluation, such as tokens, and evaluate output regardless of type.

Despite these issues, the use of Acc can be advantageous to GEC. Correcting errors in a piece of text implies reducing the initial error rate, or, in other words, increasing accuracy.[5] Therefore, Acc provides a straightforward way of comparing performance, even if the values seem inflated. If a system achieves higher Acc than a baseline, it is reducing the number of errors and thus improving the original text.

In addition, it is arguable that TNs are irrelevant and should not be considered. Unlike in information retrieval, for example, where the whole document collection is usually unknown to the user (so TNs are perhaps less relevant), the sentences fed into a GEC system are provided by users who expect all of their writing to be judged.

[2]However, Rozovskaya and Roth (2010c) propose the following formula to compute post-system error rates from an initial (base) error rate and a system's P and R:

$$\text{ErrorRate}_{sys} = \frac{\text{ErrorRate}_{base} \times (\text{P} + \text{R} - 2\text{PR})}{\text{P}}$$

[3]Considering some tokens as both a FP and a FN complicates the computation of Acc for correction. Here, we have counted such cases only once in the denominator but we propose a more elaborate solution in Section 3.2.

[4]Some relevant instances may be easy to define (e.g. for article errors, all potential noun phrases; for verb form errors, all verbs; and so on) but what are the relevant instances for idiom errors?

[5]Error rate $= 1 -$ Acc.

Aspect	P	R
Detection	$\frac{\#\text{ detected edits}}{\#\text{ spurious edits} + \#\text{ detected edits}}$	$\frac{\#\text{ detected edits}}{\#\text{ gold edits}}$
Recognition	$\frac{\#\text{ recognised edits}}{\#\text{ system edits}}$	$\frac{\#\text{ recognised edits}}{\#\text{ gold edits}}$
Correction	$\frac{\#\text{ valid corrections}}{\#\text{ system edits}}$	$\frac{\#\text{ valid corrections}}{\#\text{ gold edits}}$

Table 3.3: Definition of evaluation metrics for the HOO 2011 & 2012 shared tasks. The 'with bonus' versions include the number of missing optional edits in both the numerator and denominator for all aspects and metrics.

In this context, TNs are relevant because they indicate what parts of the text are already correct, allowing users to focus on problematic regions. Thus, Acc seems a more appropriate measure for our purposes although P, R and F are often preferred.

3.1.2 Evaluation in HOO 2011 & 2012

The HOO 2011 & 2012 shared tasks included evaluation for the following three aspects:

Detection: Whether the system determines that a correction is needed at some point in the text. A 'lenient' matching is used in this case, where a system edit is only required to have some overlap with a gold-standard edit.

Recognition: Whether the system matches the exact span of a required correction (i.e. strict alignment). Recognition in the 2012 edition also requires matching the correct error type.

Correction: Whether the system provides a correction that matches one in the gold standard.

Evaluation was carried out in terms of 'edits', which could be explicitly defined by the systems or extracted automatically by comparing a system's hypothesis to the source text. These system edits were then compared to the ones in the gold standard and classified accordingly for each of the above categories. Some edits were marked as optional in the gold standard, so systems were not required to match them although they helped to earn 'bonus' points. Unnecessary changes (i.e. FPs) were considered 'spurious' edits. More details on the datasets, alignments and edits are given by Dale and Narroway (2011). P and R were computed as shown in Table 3.3, with and without 'bonus'. F_1 (Equation 3.4) was also reported.

As noted by the organisers of the shared task, this scoring scheme has a major deficiency: if a system's edits are equivalent to the ones in the gold standard (i.e. they produce the same correction) but are composed differently (e.g. one edit versus two edits), no credit is assigned to the system. Table 3.4 illustrates this situation.

Source	I hope that these informations will be useful.
System hypothesis	I hope that this information will be useful.
System edits	(these informations → this information)
Gold edits	(these → this), (informations → information)

Table 3.4: Mismatch between system and gold standard edits producing the same corrected sentence.

3.1.3 Evaluation in CoNLL 2013 & 2014: M^2 Scorer

In order to mitigate the problems with previous approaches, a new evaluation tool was adopted for the CoNLL 2013 & 2014 shared tasks. This new tool, called MaxMatch or M^2 Scorer (Dahlmeier and Ng, 2012b)[6], employs an efficient algorithm to compute phrase-level edits and score systems based on the maximum overlap with the gold standard.

The M^2 Scorer starts by generating the best alignment between the source sentence
showing the alternative paths that convert the source sentence into the hypothesis. System edits are then extracted from this lattice according to the following criteria.

First, annotators can correct errors in different ways, often including unmodified parts of the original sentence (e.g. word → a word).[7] To allow for unmodified tokens in the edits but prevent them from spanning an unreasonable extent of the sentence, a parameter u is used to specify the maximum number of unchanged tokens allowed in an edit. By default, this parameter is arbitrarily set to 2.

Second, to solve mismatches between equivalent edits (see Table 3.4), the M^2 Scorer applies a transitive rule. This makes it possible to combine edits and maximise the chance of matching the gold standard, e.g. (ϵ → a) + (word → word) ⇒ (word → a word). The shortest maximally-matching edit sequence is computed from the lattice using an efficient search algorithm. P and R are then computed as follows:

$$\mathrm{P} = \frac{\sum_{i=1}^{n} |\mathbf{e}_\mathrm{i} \cap \mathbf{g}_\mathrm{i}|}{\sum_{i=1}^{n} |\mathbf{e}_\mathrm{i}|} \tag{3.6}$$

$$\mathrm{R} = \frac{\sum_{i=1}^{n} |\mathbf{e}_\mathrm{i} \cap \mathbf{g}_\mathrm{i}|}{\sum_{i=1}^{n} |\mathbf{g}_\mathrm{i}|} \tag{3.7}$$

where $\{\mathbf{e}_1, ..., \mathbf{e}_\mathrm{n}\}$ represent system edits and $\{\mathbf{g}_1, ..., \mathbf{g}_\mathrm{n}\}$ represent gold-standard edits. The intersection of both sets includes edits that have the same token offsets and where $\mathbf{e}_\mathrm{i}$'s correction is included in $\mathbf{g}_\mathrm{i}$. F_1 is also computed using its standard definition (Equation 3.4).

[6] Available at `http://www.comp.nus.edu.sg/~nlp/sw/m2scorer.tar.gz`

[7] These should be exceptions, since annotation guidelines should discourage the inclusion of unmodified words in an edit. Edits should include the minimum string required to correct an error but this was not the case in NUCLE.

Finally, since participating teams were allowed to submit alternative corrections, the M^2 Scorer was designed to take multiple annotations into account. In particular, for each sentence in the gold standard, the scorer evaluates its alternative sentence-level annotations and chooses the one that maximises cumulative F. This is a simple approach where each hypothesis is matched with the most similar full-sentence annotation but is unable to 'mix and match' individual cross-annotator corrections. For this reason, system performance can often be underestimated, as combining corrections from different annotations will not gain any credit.

A number of changes were introduced in the CoNLL 2014 shared task. To accommodate the notion that P is preferred over R in GEC, the β parameter in Equation 3.3 was changed from 1 to 0.5, weighting P twice as highly as R. The definition of $F_{0.5}$ is shown in Equation 3.8.

$$F_{0.5} = (1 + 0.5^2) \times \frac{P \times R}{(0.5^2 \times P) + R} \tag{3.8}$$

Additionally, the new test set included corrections from two annotators, making it the first publicly available dataset to include multiple alternative annotations.

The M^2 Scorer became the *de facto* standard for GEC evaluation, especially for comparison with previous work. It was also adopted as the official evaluation scorer in the QALB-2014 (Mohit et al., 2014) and QALB-2015 (Rozovskaya et al., 2015) shared tasks, which continued to use F_1.

However, the M^2 Scorer suffers from a number of limitations, some of which are inherent in the definitions of traditional metrics:

(a) There is a limit to the number of unchanged words allowed in an edit ($u = 2$ by default), which is arbitrary and affects final results.

(b) Given that the computed metrics rely on TP counts, a baseline system that does not propose any correct edits will not produce informative results ($P = 1$ by definition, $R = 0$ and $F = 0$). The actual error rate and consequent potential for text improvement are not taken into account.

(c) It is not possible to discriminate between a 'do nothing' baseline system and other systems that only propose wrong corrections, as they will all yield $F = 0$.

(d) System performance is underestimated when using multiple annotations for a sentence, since the scorer will choose the one that maximises F instead of mixing and matching all the available individual corrections (see Table 3.5).

(e) Partial matches are ignored (see Table 3.6).

(f) Phrase-level edits can produce misleading results, as they may not always reflect effective improvements (see Table 3.7).

Source	**Annotator 1**	**Annotator 2**		
This machines is designed for help people .	(This → These), (is → are), (help → helping)	(machines → machine), (for → to)		
System hypothesis	**System edits**	**P**	**R**	**$F_{0.5}$**
These machines are designed to help people .	(This → These), (is → are), (for → to)	0.67	0.67	0.67

Table 3.5: The M^2 Scorer is unable to mix and match corrections from different annotators.

Source	**Gold edits**			
Machine is design to help people .	(Machine → Machines), (is design → are designed)			
System hypothesis	**System edits**	**P**	**R**	**$F_{0.5}$**
Machine is designed to help people .	(design → designed)	0.00	0.00	0.00

Table 3.6: Partial matches are ignored by the M^2 Scorer.

(g) The lack of a TN count (i.e. the number of non-errors) precludes the computation of accuracy, which is useful for discriminating between systems with $F = 0$.

(h) There is no clear indicator of improvement on the original text after applying the suggested corrections, since an increase in P, R or F does not imply a reduction in the error rate (see Section 3.2.3.3).

(i) It is not clear how values of F should be interpreted (especially for $F_{0.5}$), as there is no known threshold that would signal improvement. Ranking by F does not guarantee that the top systems make the source text better. Comparing systems with the 'do nothing' baseline will not help either, since the latter will always yield $F = 0$ and make any system look better in comparison.

(j) Detection scores are not computed.

In addition, Leacock et al. (2014, p. 44) discuss key issues concerning system evaluation, such as the estimation of TNs and good practices for reporting results, which are currently not addressed by the M^2 scorer.

3.2 Towards a new evaluation method: I-measure

A better evaluation method should address the issues described above and use a measure that is meaningful and easy to interpret. In this section, we propose the I-measure,[8] a new method that aims to resolve these problems.

[8] An open-source implementation is available at `https://github.com/mfelice/imeasure`.

Source	**Gold edits**			
Machine is design to help people .	(Machine → Machines), (is → are), (design → designed)			
System hypothesis	**System edits**	**P**	**R**	**$F_{0.5}$**
The machine is designed for helping people .	(Machine is → The machine is), (design → designed), (to help people → for helping people)	0.33	0.33	0.33
Machines is a design on the helping of the people .	(Machine → Machines), (is design to help → is a design on the helping of the)	0.50	0.33	0.45

Table 3.7: The M^2 Scorer evaluates systems based on the number of edits, regardless of their length and their effect on the final corrected sentence. The first hypothesis is better than the second despite having a lower $F_{0.5}$-score.

Our proposed method uses tokens as the unit of evaluation (instead of phrase-level edits used by previous approaches), which provides a stable unit of comparison and facilitates the computation of TNs. As a result, we are able to provide a solution for problems 3.1.3.(a), 3.1.3.(e), 3.1.3.(f) and 3.1.3.(g).

The following sections describe the three pillars of our method: a new annotation scheme, sentence alignment and metrics.

3.2.1 Annotation

We define a gold standard format where each sentence is annotated with a set of errors and their possible corrections. A sentence can contain zero or more errors, each of which includes information such as a type, a flag indicating if a correction is required, and a list of alternative corrections proposed by the annotators. We consider an error requires correction if all the annotators provide a correction for it.

Unlike in other annotation schemes, each error is defined by its locus (regardless of the position of the incorrect tokens in the sentence) and all its alternative corrections must be mutually exclusive. In other words, corrections are grouped whenever they refer to the same underlying error, even if the tokens involved are not contiguous. Listing 3.1 shows a sample XML annotation for the sentence in Table 3.5 while Appendix C shows an example conversion for a sentence in the CLC (whose original annotation is shown in Figure 2.2).

Because all the correction alternatives are now mutually exclusive, we can directly combine them to generate all possible valid gold standard references. For example, the annotation in Listing 3.1 would produce the following four references:

These *machines* ***are*** *designed for* ***helping*** *people .*

These *machines* ***are*** *designed* ***to*** *help people .*

This ***machine*** *is designed for* ***helping*** *people .*

This ***machine*** *is designed* ***to*** *help people .*

```
<sentence id="1" numann="2">
  <text>
    This machines is designed for help people .
  </text>
  <error-list>
    <error id="1" req="yes" type="SVA">
      <alt ann="0">
        <c start="0" end="1">These</c>
        <c start="2" end="3">are</c>
      </alt>
      <alt ann="1">
        <c start="1" end="2">machine</c>
      </alt>
    </error>
    <error id="2" req="yes" type="Vform">
      <alt ann="0">
        <c start="5" end="6">helping</c>
      </alt>
      <alt ann="1">
        <c start="4" end="5">to</c>
      </alt>
    </error>
  </error-list>
</sentence>
```

Listing 3.1: An example annotated sentence.

By mixing and matching corrections from different annotators, we avoid the performance underestimation described in 3.1.3.(d).

3.2.2 Alignment

In order to compute matches for detection and correction, we generate a token-level alignment between a source sentence, a system's hypothesis, and a gold standard reference. Three-way alignments are a special case of *multiple sequence alignment*, a well-known string matching problem in computational biology (Mount, 2004).

We generate an exact (globally optimal) alignment using a dynamic programming implementation of the Sum of Pairs (SP) alignment (Carrillo and Lipman, 1988), shown in Listing 3.2. Under this model, the score of a multiple alignment is the sum of the scores of each pairwise alignment, so that a globally optimal alignment has minimum SP score.

Time and space complexity of the dynamic programming implementation for k strings of length n is $O(n^k)$, which is acceptable for three average-length sentences but can quickly become impractical for a larger number of sequences. In our test sets, average sentence length ranges from 15 tokens in FCE-test to 23 tokens in the CoNLL-2014 test set. Sentences in the first dataset contain an average of 2.56 errors and have been corrected by only one annotator whereas sentences in the second dataset contain an average of 4.76 errors and have two alternative annotations. This

	Initial alignment	**Desired alignment**
(S)ource	X Y	X Y
(H)ypothesis	– Z	Z –
(R)eference	X –	X –

Table 3.8: Initial and desired alignments showing differences in the placement of gaps.

poses the CoNLL-2014 test set as our worst case scenario, where each sentence produces roughly $2^5 = 32$ alignments with an average time and space complexity of $O(23^3)$ each. In practice, however, computation is typically faster, since complexity is reduced to $O(n^2)$ any time two sentences are equal (e.g. *source* = *reference*, at least 10% of cases in the CoNLL-2014 test set) and $O(n)$ when the three sentences are the same (i.e. *source* = *hypothesis* = *reference*, the trivial case).

In computational biology, edit costs are defined in terms of mutation probabilities, which are irrelevant to our task. However, we can find new optimal costs by defining a set of constraints that are meaningful for error correction:

(a) Matches have zero cost ($c_{\text{match}} = 0$).

(b) Gaps (insertions or deletions) are more costly than matches ($c_{\text{gap}} > c_{\text{match}}$).

(c) Mismatches (substitutions) are set to be more costly than gaps (insertions or deletions) so as to maximise matches ($c_{\text{mis}} > c_{\text{gap}}$).

Given these constraints, we can set $c_{\text{gap}} = 1$ and $c_{\text{mis}} = 2$; however, this will not necessarily keep gaps aligned (see Table 3.8). To ensure this, we must place a new constraint on the SP algorithm so that a gap-aligned version (desired alignment) has a lower cost than a gap-unaligned version (initial alignment):

$$(\text{X}, \text{–}) + (\text{–}, \text{X}) + (\text{X}, \text{X}) + (\text{Y}, \text{Z}) + (\text{Z}, \text{–}) + (\text{Y}, \text{–}) > (\text{X}, \text{Z}) + (\text{Z}, \text{X}) + (\text{X}, \text{X}) + (\text{Y}, \text{–}) + (\text{–}, \text{–}) + (\text{Y}, \text{–})$$

$$c_{\text{gap}} + c_{\text{gap}} + c_{\text{match}} + c_{\text{mis}} + c_{\text{gap}} + c_{\text{gap}} > c_{\text{mis}} + c_{\text{mis}} + c_{\text{match}} + c_{\text{gap}} + c_{\text{match}} + c_{\text{gap}}$$

$$4c_{\text{gap}} + c_{\text{mis}} > 2c_{\text{mis}} + 2c_{\text{gap}}$$

$$2c_{\text{gap}} > c_{\text{mis}}$$

Therefore $2c_{\text{gap}} > c_{\text{mis}} > c_{\text{gap}} > c_{\text{match}}$. For our implementation, we adopted $c_{\text{gap}} = 2$ and $c_{\text{mis}} = 3$, the minimum integer weights that fulfil this condition.

There can be more than one optimal alignment for a given set of strings. Some of these alignments will look more intuitive than others (see Table 3.9) but they are equally optimal for our evaluation method and will produce the same final results. For this reason, there is no need to extend the algorithm to find the most natural alignment.

```
/* Initialisation */
c_match := cost of match
c_mis := cost of mismatch
c_gap := cost of gap

D[0, 0, 0] := 0

D_1,2[i, j] := edit_distance(S_1[1..i], S_2[1..j])
D_1,3[i, k] := edit_distance(S_1[1..i], S_3[1..k])
D_2,3[j, k] := edit_distance(S_2[1..j], S_3[1..k])

/* Recurrences for boundary cells */
D[i, j, 0] := D_1,2[i, j] + (i + j) * c_gap,
D[i, 0, k] := D_1,3[i, k] + (i + k) * c_gap,
D[0, j, k] := D_2,3[j, k] + (j + k) * c_gap,

/* Recurrences for non-boundary cells */
for i := 1 to n_1 do
  for j := 1 to n_2 do
    for k := 1 to n_3 do
      begin
        if (S_1[i] = S_2[j]) then cij := c_match
        else cij := c_mis;
        if (S_1[i] = S_3[k]) then cik := c_match
        else cik := c_mis;
        if (S_2[j] = S_3[k]) then cjk := c_match
        else cjk := c_mis;

        d1 := D[i-1, j-1, k-1] + cij + cik + cjk;
        d2 := D[i-1, j-1, k] + cij + 2 * c_gap;
        d3 := D[i-1, j, k-1] + cik + 2 * c_gap;
        d4 := D[i, j-1, k-1] + cjk + 2 * c_gap;
        d5 := D[i-1, j, k] + 2 * c_gap;
        d6 := D[i, j-1, k] + 2 * c_gap;
        d7 := D[i, j, k-1] + 2 * c_gap;

        D[i, j, k] := Min(d1,d2,d3,d4,d5,d6,d7);
      end;
```

Listing 3.2: The Sum of Pairs dynamic programming algorithm for the alignment of three sequences, S_1, S_2 and S_3 (adapted from Gusfield (1997)).

3.2.3 Metrics

Once we have an optimal alignment between a source, a hypothesis and a reference, we can collect the necessary counts for our metrics. The limitation in 3.1.3.(j) is addressed by computing these metrics for both detection and correction.

We adopt an extended version of the WAS evaluation scheme (see Table 3.1) where each token alignment is classified as a TP, TN, FP or FN. As noted by Chodorow et al. (2012), cases where *source* $\neq$ *hypothesis* $\neq$ *reference* are both a FP and a FN for correction, so we introduce a new FPN class to count such cases and adjust our metrics accordingly. Our extended WAS scheme is shown in Table 3.10.

With these counts, we can compute token-level P, R and F_β using their standard definitions. However, as mentioned in Section 3.1.3, the F-measure does not shed

S	Their	is	wide	spread	usage	of	technology	.		A	A	A	A	A	A	A	A
H	There	is	widespread	use		of	technology	.	⇔	B	A	B	B	-	A	A	A
R	There	is	widespread	use		of	technology	.		B	A	B	B	-	A	A	A
S	Their	is	wide	spread	usage	of	technology	.		A	A	A	A	A	A	A	A
H	There	is	widespread		use	of	technology	.	⇔	B	A	B	-	B	A	A	A
R	There	is	widespread		use	of	technology	.		B	A	B	-	B	A	A	A

Table 3.9: Two equally optimal alignments under the SP alignment model.

Tokens			**Classification**	
Source	**Hypothesis**	**Reference**	**Detection**	**Correction**
a	a	a	TN	TN
a	a	b	FN	FN
a	a	-	FN	FN
a	b	a	FP	FP
a	b	b	TP	TP
a	b	c	TP	FP, FN, FPN
a	b	-	TP	FP, FN, FPN
a	-	a	FP	FP
a	-	b	TP	FP, FN, FPN
a	-	-	TP	TP
-	a	a	TP	TP
-	a	b	TP	FP, FN, FPN
-	a	-	FP	FP
-	-	a	FN	FN

Table 3.10: Our extended WAS evaluation scheme.

light on the error rates in the data and is unable to discriminate between a 'do nothing' baseline and other systems unless TP > 0. Because we now have a TN count, we can compute Acc instead as in Equation 3.9 and thus solve problems 3.1.3.(b) and 3.1.3.(c).

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN} - \text{FPN}} \tag{3.9}$$

Since we consider TNs relevant for GEC, accuracy seems a more appropriate measure of correction quality than F-score.

3.2.3.1 Weighted accuracy

Acc treats all counts equally, which has two main side effects. First, a system that introduces the same number of TPs and FPs will have the same accuracy as the 'do nothing' baseline, in which case we would prefer to keep the original text and rank the system lower, in accord with the choice of $F_{0.5}$ in the CoNLL 2014 shared task. Second, Acc is also unable to discriminate between systems with different TP and TN counts if their sum is the same, e.g. $\text{TP} = 4, \text{TN} = 6$ vs $\text{TP} = 7, \text{TN} = 3$.

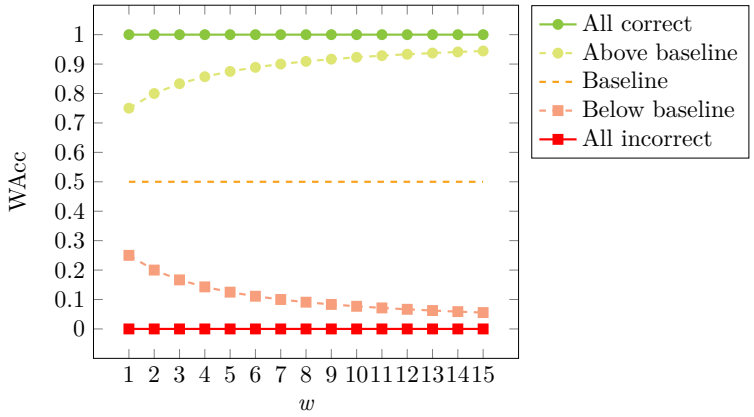


Figure 3.1: Effect of w on WAcc.

It is clear that for error correction these counts should be weighted differently. In particular, we would like to:

- Reward correction more than preservation (i.e. $weight_{\mathrm{TP}} > weight_{\mathrm{TN}}$).
- Penalise unnecessary corrections more than uncorrected errors (i.e. $weight_{\mathrm{FP}} > weight_{\mathrm{FN}}$).

We can reformulate Acc to satisfy these conditions by including a weight factor $w > 1$, leading to weighted accuracy (WAcc):

$$\mathrm{WAcc} = \frac{w \cdot \mathrm{TP} + \mathrm{TN}}{w \cdot \mathrm{TP} + \mathrm{TN} + w \cdot \left(\mathrm{FP} - \frac{\mathrm{FPN}}{2}\right) + \left(\mathrm{FN} - \frac{\mathrm{FPN}}{2}\right)} \tag{3.10}$$

$$= \frac{w \cdot \mathrm{TP} + \mathrm{TN}}{w \cdot \mathrm{TP} + \mathrm{TN} + w \cdot \mathrm{FP} - w \cdot \frac{\mathrm{FPN}}{2} + \mathrm{FN} - \frac{\mathrm{FPN}}{2}} \tag{3.11}$$

$$= \frac{w \cdot \mathrm{TP} + \mathrm{TN}}{w \cdot (\mathrm{TP} + \mathrm{FP}) + \mathrm{TN} + \mathrm{FN} - (w+1) \cdot \frac{\mathrm{FPN}}{2}} \tag{3.12}$$

Note that FPN cases are now distributed equally between FPs and FNs, so that the introduction of w will not affect their proportions.

Higher values of w will reward and penalise systems more heavily, bringing those below the baseline closer to the lower bound and those above the baseline closer to the upper bound (see Figure 3.1). As w increases, differences between $\mathrm{WAcc}_{\mathrm{sys}}$ and its bounds become less pronounced, which is why we adopt $w = 2$ (the minimum integer value that satisfies the condition). Regardless of w, WAcc will always reduce to Acc for the 'do nothing' baseline.

3.2.3.2 Metric behaviour

Before we set out to evaluate and compare systems, we must understand how metrics behave and to what extent they are comparable.

System	Chosen references	P	R	$F_{0.5}$
S_1	**1.2, 2.1, 3.1**	**0.60**	**0.20**	**0.43**
S_2	1.2, 2.1, 3.1	0.80	0.05	0.20
S_1	1.1, 2.1, 3.2	0.30	0.30	0.30
S_2	**1.1, 2.1, 3.2**	**0.30**	**0.40**	**0.32**

Table 3.11: S_1 outperforms S_2 in terms of overall $F_{0.5}$ but S_2 outperforms S_1 when evaluated on different references.

System	TP	FP	TN	FN	P	R	$F_{0.5}$	Acc	WAcc
Baseline	0	0	6	4	1.00	0.00	0.00	0.60	0.60
S_1	4	1	5	0	0.80	1.00	0.83	0.90	0.87
S_2	1	0	6	3	1.00	0.25	0.62	0.70	0.73
S_3	1	1	5	3	0.50	0.25	0.42	0.60	0.58
S_4	4	6	0	0	0.40	1.00	0.45	0.40	0.40

Table 3.12: An increase in P, R or F does not necessarily translate into an increase in Acc, assuming all systems are evaluated on the same set of references.

Table 3.10 indicates that the metrics will always produce the same results for detection and correction unless *source* $\neq$ *hypothesis* $\neq$ *reference* for at least one position in the alignment. A 'do nothing' baseline will always produce the same results for both aspects, since *source* = *hypothesis* for all positions.

Whenever a gold standard allows for alternative corrections, references that maximise the target metric should be chosen. Nevertheless, we note that the (maximum) score obtained by a system only applies to a given set of chosen references and is therefore only directly comparable to results on the same reference set. To illustrate this, consider two systems (S_1 and S_2) evaluated on a gold standard containing 3 sentences with 2 correction alternatives each (i.e. six possible references: 1.1, 1.2, 2.1, 2.2, 3.1 and 3.2 respectively). Table 3.11 shows that, while S_1 achieves a higher maximum score than S_2, comparing their $F_{0.5}$ scores directly is not possible as they are computed on a different set of references. In fact, S_2 could outperform S_1 on other reference sets.

3.2.3.3 Measuring improvement

As noted in Section 3.1.1, whenever $P > 0.5$, the error rate decreases (and therefore Acc increases) so the text should be improved.[9] However, an increase in P, R or F alone does not necessarily imply an increase in Acc or WAcc, as illustrated in Table 3.12.

In order to determine whether a system improves the source text, we must compare its performance ($WAcc_{sys}$) with that of the 'do nothing' baseline ($WAcc_{base}$). Since each $WAcc_{sys}$ is computed from a different set of references, we must compute

[9]The $P > 0.5$ criterion also holds for Acc but not WAcc, as the latter modifies the original proportions by introducing weights.

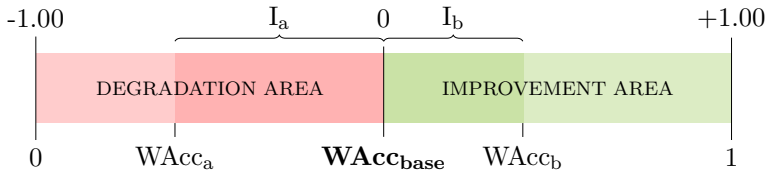


Figure 3.2: Graphical representation of improvement for two hypothetical systems, a and b. Values of I are shown at the top while values of WAcc are shown at the bottom.

$\text{WAcc}_{\text{base}}$ individually for each system using its chosen references. This is done by using the source sentence as the hypothesis in the existing alignment. Once we have WAcc_{sys} and $\text{WAcc}_{\text{base}}$ for each system, we can compare them to determine if the text has improved. When these two values are equal, there is no benefit to deploying the system.

If we want to compare and rank systems, we need to measure how much the text has been improved or degraded. This can be done using a baseline-normalised metric that measures relative coverage of the area between the baseline and WAcc bounds (see Figure 3.2). This metric, henceforth *Improvement* or I-measure (I), is defined as:

$$\text{I} = \begin{cases} \lfloor \text{WAcc}_{\text{sys}} \rfloor & \text{if } \text{WAcc}_{\text{sys}} = \text{WAcc}_{\text{base}} \\ \dfrac{\text{WAcc}_{\text{sys}} - \text{WAcc}_{\text{base}}}{1 - \text{WAcc}_{\text{base}}} & \text{if } \text{WAcc}_{\text{sys}} > \text{WAcc}_{\text{base}} \\ \dfrac{\text{WAcc}_{\text{sys}}}{\text{WAcc}_{\text{base}}} - 1 & \text{otherwise} \end{cases} \tag{3.13}$$

where $\lfloor x \rfloor$ stands for the *floor* function. Values of I lie in the $[-1, 1]$ interval and should be interpreted as per Table 3.13. The use of this metric provides a solution to problems 3.1.3.(h) and 3.1.3.(i).

The idea of measuring performance in terms of degradation or improvement has also been explored by Wagner (2012), although his method is radically different. While he compares classifiers by representing TP and TN rates on a plane, we provide a one-dimensional measure that compares a system with a baseline directly using a weighted version of Acc.

The I-measure should be computed after maximising system WAcc at the sentence level, so as to ensure all the evaluated hypotheses are paired with their highest scoring references. Trying to maximise I 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).

To facilitate reading, we report all measures as percentages from now on, so that I-measure scores range within $[-100, 100]$ and all the other metrics within $[0, 100]$.

Value	Interpretation
1	100% improvement (100% correct text).
> 0	Relative improvement.
0	Baseline performance (no change).
< 0	Relative degradation.
-1	100% degradation (100% incorrect text).

Table 3.13: Interpretation of I values.

S:	Staff in		hospital	can help rule out		serious medical	condition	.
R_1:	Hospital		staff	can help rule out	a	serious medical	condition	.
R_2:	Staff in	a	hospital	can help rule out	any	serious medical	condition	.
R_3:	Staff in		hospitals	can help rule out		serious medical	conditions	.

Figure 3.3: Automatic grouping of overlapping corrections into error clusters.

3.2.4 Experiments and results

We tested our evaluation method by re-ranking systems in the CoNLL 2014 shared task on grammatical error correction. Re-ranking was limited to the 12 participating teams that made their system's output publicly available.

For the gold standard, we used the shared task test set containing corrections from the two official annotators as well as alternative corrections provided by three participating teams. This version allowed us to generate many more references than the original test set and thus reduce annotator bias.

In order to mix and match corrections from different annotators, we had to cluster the original shared task annotations into groups of independent errors and mutually exclusive correction alternatives. We applied an automatic process based on token overlap, assuming that overlapping corrections from different annotators are mutually exclusive (i.e. you can only apply one of them at a time). Figure 3.3 shows an example where the phrase *Staff in hospital* has been corrected in three different ways: *Hospital staff*, *Staff in a hospital* or *Staff in hospitals*. Because these corrections overlap by at least one token, they are assumed to fix the same underlying error and be mutually exclusive, so that illegal combinations such as **Staff in* ***a hospitals*** are not allowed.

This natural grouping of corrections into 'clusters' (the *error* elements in Listing 3.1) reveals the number of independent errors in the sentence, so we can mix and match corrections from each cluster safely to generate all valid grammatical references. Unfortunately, some corrections cannot be grouped by token overlap and require human intervention to determine if they are mutually exclusive. For the example above, our automatic method would create three clusters, as it would not know that *a*, *any* and *conditions* are alternative corrections for the same agreement error and should be grouped into one single cluster (there is no overlap between them).

System	Original annotations			Mixed annotations		
	P	**R**	**$F_{0.5}$ ↓**	**P**	**R**	**$F_{0.5}$ ↓**
CUUI	47.66	33.87	44.07	47.57	39.60	45.73
AMU	44.68	29.44	40.48	44.56	33.49	41.80
CAMB	39.22	41.65	39.69	39.04	48.72	40.66
POST	36.39	29.13	34.67	36.39	33.79	35.84
NTHU	33.56	28.10	32.31	33.62	31.52	33.18
UMC	34.86	20.86	30.73	34.86	23.31	31.71
RAC	33.67	19.08	29.21	33.67	21.59	30.28
PKU	32.17	19.60	28.51	32.42	21.63	29.48
SJTU	28.00	7.08	17.60	28.00	7.46	18.06
UFC	73.08	3.26	13.83	73.08	3.39	14.31
IPN	9.16	3.87	7.20	9.16	4.09	7.34
IITB	30.30	1.74	7.07	30.30	1.81	7.31
BASELINE	100.00	0.00	0.00	100.00	0.00	0.00

Table 3.14: M^2 Scorer results (in percentages).

System	TP	TN	FP	FN	FPN	P	R	$F_{0.5}$	Acc	WAcc	$WAcc_{base}$	I ↓
UFC	19	13,062	7	665	2	73.08	2.78	12.06	95.13	95.09	95.03	1.35
BASELINE	0	13,078	0	673	0	100.00	0.00	0.00	95.11	95.11	95.11	0.00
IITB	11	13,057	26	668	4	29.73	1.62	6.65	94.98	94.82	95.06	−0.25
SJTU	54	12,947	114	649	8	32.14	7.68	19.64	94.51	93.79	94.89	−1.16
CUUI	290	12,697	337	553	34	46.25	34.40	43.27	93.82	91.86	93.91	−2.18
PKU	128	12,800	283	625	66	31.14	17.00	26.70	93.89	92.28	94.53	−2.38
AMU	219	12,761	322	556	41	40.48	28.26	37.26	93.94	92.06	94.39	−2.47
UMC	179	12,761	314	603	26	36.31	22.89	32.50	93.56	91.67	94.35	−2.84
IPN	25	12,848	251	680	40	9.06	3.55	6.91	93.53	92.00	94.88	−3.04
POST	231	12,588	454	574	46	33.72	28.70	32.58	92.88	90.23	94.17	−4.18
RAC	147	12,723	426	623	49	25.65	19.09	24.00	92.79	90.28	94.45	−4.41
CAMB	386	12,402	641	502	78	37.59	43.47	38.63	92.31	88.77	93.59	−5.15
NTHU	196	12,620	521	575	54	27.34	25.42	26.93	92.48	89.44	94.44	−5.29

Table 3.15: Results of our new evaluation method (in percentages). All values of I are statistically significant (two-tailed paired T-test, $p < .01$). Rankings by I and $F_{0.5}$ are clearly distinct.

We developed a few heuristics to determine whether a sentence is likely to require human intervention, for example if it has clusters containing corrections from only a subset of the annotators (as in our previous example). These sentences were excluded from the test set, leaving a subset of 711 sentences out of 1,312 (54.19%).

We restrict our analysis to correction, since that is the only aspect reported by the M^2 Scorer. Table 3.14 shows the results of the M^2 Scorer using the original annotations as well as a modified version containing mixed-and-matched corrections. Results of our proposed evaluation method are included in Table 3.15.

As expected, rankings are clearly distinct between the two methods, as they use different units of evaluation (phrase-level edits vs. tokens) and maximising metrics ($F_{0.5}$ vs. WAcc). Results show that only the UFC system is able to beat the baseline (by a small but statistically significant margin), being also the one with consistently highest P (much higher than the rest).

These rankings are affected by the fact that systems were probably optimised for $F_{0.5}$ during development, as it was the official evaluation metric for the shared task. Rankings by $F_{0.5}$ are almost identical for the two methods (Spearman's rank correlation is 0.9835 with $p < .01$), suggesting that there is a statistically significant relationship between phrase-level edits and tokens, especially since phrases are only 1.12 tokens on average in this dataset.

Spearman's ρ between both scorers ($F_{0.5}$ vs. I) is -0.5330, which suggests they generally produce inverse rankings. Pearson's correlation between token-level $F_{0.5}$ and I is -0.5942, confirming the relationship between rankings and our intuition that $F_{0.5}$ is not a good indicator of overall correction quality. While I reflects improvement, $F_{0.5}$ indicates error manipulation. We argue that I is better suited to the needs of end-users (as it indicates whether the output of the system is better than the original text) whereas $F_{0.5}$ is more relevant to system developers (since they need to analyse P and R in order to tune their systems).

Lastly, we verify that mixing and matching corrections from different annotators improves R (see Table 3.14) and gives a better estimate of true performance.

3.3 New directions: crowdsourced evaluation

The use of automatic metrics to evaluate systems is standard practice in NLP; however, they perform poorly in comparison with humans for text generation tasks such as machine translation, summarisation or error correction, where many answers can be valid. This fact has been recognised by the MT community, who conduct manual evaluation of systems' output in their annual workshops, but was not properly considered in GEC until recently.

There are many aspects in which automatic evaluation falls behind human judgements. Madnani et al. (2011), for example, observed that automatic metrics are based on two strict categories ('error' and 'OK'), when in reality grammatical acceptability should be modelled as a continuum based on the proportion of human raters that judge an instance as correct or incorrect. To prove their point, the authors conducted an experiment on the detection of extraneous prepositions. They compiled a corpus of 1,000 sentences (of which 500 had an extraneous preposition) and hired 20 anonymous workers on a crowdsourcing website to mark the errors.

The following weighted versions of 'hits', 'misses' and FPs were proposed:

$$\mathrm{H}_w = \sum_i^N (c^i_{\mathrm{sys}} * p^i_{\mathrm{crowd}}) \tag{3.14}$$

$$\mathrm{M}_w = \sum_i^N ((1 - c^i_{\mathrm{sys}}) * p^i_{\mathrm{crowd}}) \tag{3.15}$$

$$\mathrm{FP}_w = \sum_i^N (c^i_{\mathrm{sys}} * (1 - p^i_{\mathrm{crowd}})) \tag{3.16}$$

where N is the total number of instances, c^i_{sys} is the class assigned by the system (0 = OK, 1 = error) and p^i_{crowd} is the proportion of human raters that classified instance i as an error. Weighted versions of P and R are then computed as:

$$\mathrm{P}_w = \frac{\mathrm{H}_w}{\mathrm{H}_w + \mathrm{FP}_w} \tag{3.17}$$

$$\mathrm{R}_w = \frac{\mathrm{H}_w}{\mathrm{H}_w + \mathrm{M}_w} \tag{3.18}$$

Two ad hoc detection systems were created and evaluated using the traditional (unweighted) and weighted metrics. Results showed that the unweighted metrics overestimated performance as they counted one whole point even for cases with low agreement. On the contrary, weighted metrics gave less credit for contentious cases, providing a fairer score. The new metrics were also said to be more stable, as a change in the proportion of ratings does not have as big an impact as in the discrete (unweighted) case.

As noted earlier, the use of datasets with only one annotation is a major limitation in GEC and other text generation tasks, where there is often more than one correct answer. To assess the effect of multiple annotations on evaluation, Bryant and Ng (2015) compiled a new version of the CoNLL-2014 test data that included corrections from 10 different human annotators. Experiments confirmed the intuition that system performance increases as more annotations are added to the gold standard, doubling $\mathrm{F}_{0.5}$ performance when moving from 1 to 9 annotators.

The authors also observed high variability between the annotations and questioned the role of inter-annotator agreement in GEC, where low values are not indicative of real disagreement but rather the result of multiple valid answers. To validate these claims, each of the 10 annotations was evaluated individually using the remaining 9 as the set of references. The average of these individual results gave a realistic upper bound of $\mathrm{F}_{0.5} = 72.58$ on the task, far from the ideal $\mathrm{F}_{0.5} = 100$. In light of this result, Bryant and Ng (2015) propose scoring systems against the average human performance instead of the theoretical maximum score, so that:

$$\mathrm{Ratio} = \frac{\mathrm{F}^{\mathrm{system}}_{0.5}}{\mathrm{F}^{\mathrm{human}}_{0.5}} \tag{3.19}$$

Their study concludes that the apparent disagreement between annotators is actually the result of each annotator's bias towards a particular type of correction, which also varies per error type (i.e. some error categories enjoy more agreement than others).

Napoles et al. (2015) used crowdsourced judgements to produce a human ranking of systems in the CoNLL 2014 shared task. Three native English speakers were asked to rank hypotheses from the 12 participating systems plus the source and reference sentences, from which pairwise judgements were later derived. The generated human ranking was found to be considerably different from the official one produced by the M^2 Scorer, as well as rankings by other metrics such as the Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) or I-measure.

The authors then proposed a new evaluation measure that aims to maximise correlation with human rankings. This new method, called Generalized Language Evaluation Understanding (GLEU),[10] is a modified version of the BLEU metric widely used in MT, which computes n-gram precision over multiple references. Similar to our I-measure, GLEU rewards TPs but penalises FNs (rather than FPs). The definition of the metric is as follows:

$$p'_n = \frac{\sum_{n\text{-}gram \in C} \text{Count}_{R \setminus S}(n\text{-}gram) - \lambda(\text{Count}_{S \setminus R}(n\text{-}gram)) + \text{Count}_R(n\text{-}gram)}{\sum_{n\text{-}gram' \in C'} \text{Count}_S(n\text{-}gram') + \sum_{n\text{-}gram \in R \setminus S} \text{Count}_{R \setminus S}(n\text{-}gram)} \tag{3.20}$$

$$\text{Count}_B(n\text{-}gram) = \sum_{n\text{-}gram' \in B} d(n\text{-}gram, n\text{-}gram') \tag{3.21}$$

$$d(n\text{-}gram, n\text{-}gram') = \begin{cases} 1 & \text{if } n\text{-}gram = n\text{-}gram' \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-c/r)} & \text{if } c \leq r \end{cases} \tag{3.23}$$

$$\text{GLEU}(C, R, S) = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p'_n\right) \tag{3.24}$$

where N indicates the maximum n-gram order (4 by default), λ is a penalising factor for incorrectly changed n-grams (0.1 or 0 in their experiments), $w_n = 1/N$ and C, R and S stand for the set of correction hypotheses, references and source sentences respectively. BP is a *brevity penalty* factor that penalises hypotheses that are considerably shorter than their references, as a way of controlling for recall. Experiments with $\lambda = 0$ (GLEU_0) and $\lambda = 0.1$ ($\text{GLEU}_{0.1}$) show the former is better correlated with human rankings, although it achieves a moderate Pearson's $r = 0.542$ and Spearman's $\rho = 0.555$. Although GLEU seems the best correlated metric for GEC (at least on the evaluated dataset), it was found that all metrics deviate from human rankings, which is perhaps caused by the fact that all error types are weighted equally.

[10] Available at `https://github.com/cnap/gec-ranking/`

Rank	Napoles et al. (2015)	Grundkiewicz et al. (2015)
1	CAMB	AMU
2	AMU	RAC
3	RAC	CAMB
4	CUUI	CUUI
5	*BASELINE*	POST
6	POST	UFC
7	UFC	PKU
8	SJTU	UMC
9	IITB	IITB
10	PKU	SJTU
11	UMC	*BASELINE*
12	NTHU	NTHU
13	IPN	IPN

Table 3.16: Differences in human rankings for systems in the CoNLL 2014 shared task. *BASELINE* represents the source (uncorrected) text.

In a similar study, Grundkiewicz et al. (2015) also produced a human ranking for systems in the CoNLL 2014 shared task. The method used to collect the human judgements was similar to that of Napoles et al. (2015), with differences in the sampling method and final ranking algorithm. Inter-annotator agreement was found to be 0.29 (weak) while intra-annotator agreement was 0.46 (moderate). This is in line with discoveries by Rozovskaya and Roth (2010a), who also report low inter-annotator agreement (0.16 to 0.40) for sentences that were initially corrected by one annotator and later revised by a second annotator.

The human ranking reported by the authors differs from the one in Napoles et al. (2015) (see Table 3.16), as does the correlation of the M^2 Scorer and I-measure with this new ranking. Out of all the tested evaluated methods, the M^2 Scorer was the best correlated, although correlation was moderate. Varying the value of β in the range [0.1, 1] reveals that maximum correlation with human rankings is achieved for $\beta = 0.18$, weighting P much more than in the latest $F_{0.5}$. Finally, MT metrics were found to correlate negatively, which is surprising given the fact that they are oriented towards precision.

3.4 Statistical significance

If we aim to adhere to good research practices, system evaluations should include significance tests; however, this is not a common practice in the GEC literature. As in other branches of NLP, this is due to the fact that test sets are often small and evaluation measures rarely follow a normal distribution so common statistical tests, such as t-tests, are not appropriate.

This problem could be overcome with the use of non-parametric methods (e.g. Wilcoxon's signed-rank test) but evaluation measures such as F often have too

complex a form for these tests. Instead, researchers have turned to alternative methods that calculate significance using multiple sampling on the test set and are applicable to practically any evaluation measure. The most common methods are *randomisation tests* and the *bootstrap* (Efron and Tibshirani, 1993), which have been found to largely agree with each other (Smucker et al., 2007). For our experiments, we adopt the bootstrap method, as it seems more intuitive and particularly suited for small datasets and complex measures (Adèr et al., 2008, p. 373). The bootstrap has been successfully used with a variety of measures in different NLP tasks, including mean average precision for information retrieval (Smucker et al., 2007), BLEU for MT (Berg-Kirkpatrick et al., 2012; Koehn, 2004) and many others (Berg-Kirkpatrick et al., 2012). It is also recommended for system comparison in the GEC literature (Leacock et al., 2014, p. 43) and is already being used by researchers in the field (Grundkiewicz et al., 2015).

Assuming we have a test set with n instances that is a representative sample of the population and two systems, A and B, that we want to compare, we first evaluate both systems on the test set using the chosen evaluation measure and calculate the difference between them. This is our *observed difference*. Next, we randomly draw n instances from our test set with replacement, test both systems on this new data and store the difference. This process is repeated a number of times (typically 1,000) until we obtain a distribution of differences. For a 95% confidence interval (i.e. $\alpha = .05$), we ignore the lowest and highest 2.5% of the values in the distribution. There are now three different criteria we can use to determine significance:

1. We can check if our confidence interval includes 0, in which case the observed difference is not statistically significant.
2. We can shift the distribution of differences so that its mean is zero, thus simulating the null distribution. If the observed difference is now less than the lower bound of the new confidence interval or greater than its upper bound, the observed difference is statistically significant.
3. Using the same simulated null distribution, we can compute a *p-value* as the fraction of instances that have an absolute value equal to or greater than our observed difference. If *p-value* $\leq \alpha$, the null hypothesis is rejected and we conclude that the observed difference is statistically significant.

Statistical tests reported in this thesis followed the described procedure, using a 95% confidence interval and 1,000 iterations.

3.5 Analysis and discussion

A range of methods have been proposed to evaluate GEC systems but no entirely satisfactory method has emerged as yet. Human rankings have been used as 'ground

truth' but the variability between annotators shows that there are often many valid answers. In fact, even the human rankings generated independently by Napoles et al. (2015) and Grundkiewicz et al. (2015) do not match (Spearman's $\rho = 0.8187$, see Table 3.16). According to the former, only 4 systems improve the original text while according to the latter, this number rises to 10 (more than double). This reveals that attempts to correlate automatic evaluation metrics with human rankings seem forced, especially when inter-annotator agreement in the data used to generate these rankings is already too low (e.g. $\rho = 0.29$ reported by Grundkiewicz et al. (2015)). It would then seem inappropriate to claim that a metric which maximises correlation with these judgements is objectively more accurate.

Our proposed method (I-measure) has been questioned for its low correlation with human rankings; however, this is an inappropriate comparison, since the I-measure has not been tuned to any particular dataset. By contrast, the best correlated metrics (GLEU_0 and the M^2 Scorer with $\beta = 0.18$) have been optimised to maximise correlation with human judgements on the CoNLL 2014 test set.

The I-measure can also be optimised by tuning the weights w_i for each count in Equation 3.12 or generalising the equation to use individual weights for each of the counts, as shown below:

$$\text{WAcc} = \frac{w_1 \cdot \text{TP} + w_2 \cdot \text{TN}}{w_1 \cdot \text{TP} + w_2 \cdot \text{TN} + w_3 \cdot \left(\text{FP} - \frac{\text{FPN}}{2}\right) + w_4 \cdot \left(\text{FN} - \frac{\text{FPN}}{2}\right)} \tag{3.25}$$

As noted by Grundkiewicz et al. (2015), the inclusion of TNs in the I-measure makes it a conservative metric so unless changes are licensed in the gold standard, they will be heavily penalised. This explains the radical change in rankings between our metric and the rest. At the same time, this ensures that top-ranked systems really make the source text better, which we argue should be the ultimate goal of GEC.

Our evaluation method overcomes many of the limitations of previous approaches by using a stable unit of evaluation, weighting edit operations in line with the goals of GEC and making the most of the available annotations. Values of F are always positive, with no clear interpretation or threshold that would indicate improvement of the original text whereas the I-measure provides a meaningful indicator by analysing post-system error rate ($\text{I} < 0$ for degradation, $\text{I} = 0$ for no change and $\text{I} > 0$ for improvement). We also combine individual corrections from different annotators, as this improves R and ensures systems get more accurate scores from the available annotations.

Our experiments show that I and $\text{F}_{0.5}$ are inversely correlated and account for different aspects of system performance. Choosing one metric over the other poses a fundamental question about the aims of error correction, whether we prefer a system that tackles few errors but improves the original text or one that handles many more errors but degrades the original. We believe that, from a user perspective, a system that reliably improves the text is more desirable.

System hypotheses		Best	
		$F_{0.5}$	I
a.	The son was died after one year 's treatment and a couple got divorced later after that .	×	
b.	The son had died after one year 's and the couple got divorced later after that .		×
a.	Although there might be a lot of challenges along the way in seeking medical attention , such as a financial issues , everyone should be given right of knowing their family 's inherented medical conditions .	×	
b.	Although there might be a lot of challenges along the way in seeking medical attention , such as finance , everyone should be given the right of knowing their family 's inherented medical conditions .		×
a.	Taking Angeline Jolie , for example , she is famous but she still reveal the truth about her genetic testing to the development of her breast cancer risk .	×	
b.	Taking Angeline Jolie for example , she is famous but she still revealed the truth about her genetic testing on the development of her breast cancer risk .		×

Table 3.17: Example hypotheses produced by two error correction systems (a and b). The last two columns indicate the highest-scoring hypothesis from each pair according to each evaluation metric.

		GLEU	I
Source	Can a elephant live without tusks ?	48.10	0.00
Hypothesis 1	live without tusks ? Can an elephant	53.73	−69.89
Hypothesis 2	Can a elephant without tusks live ?	0.00	−41.67
Hypothesis 3	Giraffes are in danger of extinction .	0.00	−100.00
Reference	Can an elephant live without tusks ?	100.00	100.00

Table 3.18: Differences between GLEU and I-measure (in percentages).

Table 3.17 shows a few examples where the M^2 Scorer differs from our method, revealing how the I-measure is able to pick hypotheses in accord with (at least our) intuitions. For a detailed review of the problems with earlier metrics, refer to Section 3.1.3.

Metrics such as GLEU are not without problems either. By treating each sentence as a bag of n-grams, the enforcement of grammaticality is somewhat diluted, since points are given for matching n-grams without much consideration for their position in the sentence. As a result, GLEU seems fairly insensitive to word order, as illustrated by Hypothesis 1 in Table 3.18. It seems also unable to discriminate between incorrect sentences that have significant overlap with the source and completely random text (see Hypothesis 2 and 3 in Table 3.18).

Finding a single metric that solves all problems is elusive, since most metrics work well in some cases but fail in others. Ideally, we should be able to achieve a balance but finding independent evidence to support one correction over another is also difficult, since the notion of sentence quality is somewhat subjective (Bryant and Ng, 2015). Sentences can be corrected in many different ways and the fact that a given correction is not matched by any of the references does not necessarily mean that it is not valid. Therefore, we must accept that any metric used in such scenarios will not be perfect.

Automatic evaluation metrics that are based on comparisons with a gold standard are essentially distance metrics and so are inherently limited by the number of available references. It is then naive to expect that they will correlate well with human judgements which are based on free interpretations of quality, as the metrics will always be constrained to the given set of references.

Judging hypotheses without looking at the source or reference sentences is a distinct task, more similar to *sentence quality estimation* for MT output (Blatz et al., 2004; Specia et al., 2009). This could be potentially useful for GEC as it would ameliorate the issue that any set of gold standard references will underspecify the set of possible corrections. Meanwhile, we strongly believe that the proposed I-measure can be a suitable metric for GEC, provided that the gold standard includes a representative set of references.

Given that there is no consensus on evaluation, we report the three main evaluation metrics (M^2 Scorer, I-measure and GLEU) in all our forthcoming experiments. This will allow us to view system performance from different perspectives and develop a clearer notion of metric behaviour.

Chapter 4

Experiments on constrained error correction

This chapter describes our approach to AEG and presents a first series of experiments on a limited number of error types. The first section discusses the role of artificial errors as training data. The second and third sections describe experiments using random and probabilistic methods in the context of the CoNLL-2013 shared task.

4.1 Rationale

As explained in Section 2.2.3, artificial errors can be used as a replacement for real learner errors in cases where such data is insufficient or unavailable. This is particularly important for SMT, which requires a large amount of parallel data to generate efficient translation models and, unlike classifiers, cannot be trained on error-free data alone. Artificial errors have been shown to improve error correction performance, especially when used in conjunction with genuine error-annotated learner data (Foster and Andersen, 2009). In addition, they have been found to be more useful than error-free text (Rozovskaya and Roth, 2010c).

The biggest challenge in AEG is how to generate errors that resemble real-life learner errors which look plausible in their given context. In principle, this means that we should avoid generating random errors but focus on observed patterns and distributions, in the hope that they will improve performance. The experiments in this thesis aim to investigate this hypothesis.

It can be argued that generating realistic errors is just as hard as correcting them, since the effort and knowledge required to generate artificial errors could well be used to build a correction system instead. However, building an efficient error correction system requires not only a set of error correction patterns but additional complex knowledge about when and how to apply them. In this regard, ML approaches (and SMT in particular) provide a framework for learning such models automatically from a collection of samples, so investing in data seems a sensible decision.

Dataset	# sentences	# tokens
NUCLE	57,151	1,161,567
EVP	18,830	351,517
CoNLL-2013 test	1,381	29,207

Table 4.1: Datasets used in our random generation experiments.

AEG has two main advantages. First, it provides an economic and efficient way of producing error-tagged data, which would otherwise require manual annotation and can be difficult to obtain. Second, it allows us to control variables such as genre, error types, distributions, topic, text complexity, etc. However, as summarised in Section 2.2.3, the effect of artificial errors on system performance has not always been consistent so we aim to investigate this further throughout the rest of this thesis.

In this chapter, we describe a series of early experiments on a limited set of error types using data from the CoNLL-2013 shared task which aim to test some initial intuitions. These include random (Section 4.2) and probabilistic generation (Section 4.3). The insights gained in this chapter are used to guide our second round of experiments, following the most effective methods and promising paths. It must be noted that each experiment is self-contained, so direct comparisons between results are not always possible.

4.2 Random generation

The most basic approach to generating artificial errors is to inject them at random in error-free text. In order to test the usefulness of this method, we performed a series of experiments using data from the CoNLL-2013 shared task where we tried to augment the original training corpus (NUCLE) with random artificial errors. The CoNLL-2013 shared task focused on the correction of only five error types: articles and determiners (ArtOrDet), noun number (Nn), prepositions (Prep), subject-verb agreement (SVA) and verb form errors (Vform). As our *reference corpus*, we used a version of NUCLE containing only these error types.

To minimise the effect of genre and style on the generated data, we used error-free sentences written by learners as our *base texts* (i.e. the texts where we will inject errors). This data is a publicly available portion of the CLC, formed by all the corrected learner sentences featured on the English Vocabulary Profile (EVP) website.[1] These sentences were produced by learners with different backgrounds and levels of proficiency sitting a variety of ESOL examinations. The size of each dataset used in our experiments is given in Table 4.1.

In general, our approach to AEG is based on the extraction of errors and their corrections from a reference corpus. Each error and correction is extracted using a context window of 0, 1 or 2 tokens to the left and right, and added to a list of

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

Type	Pattern	Example
Lexical	has → have	*temperature* ***has*** *risen →* *temperature* ***have*** *risen*
	to be used → to be use	*technology to be* ***used*** *→* *technology to be* ***use***
	during → for	***during*** *the early 60s →* ***for*** *the early 60s*
PoS	NN → NNS	*information → information****s***
	DT NNP → NNP	***the*** *US → US*
	NN VBZ VBN → NN VBP VBN	*expenditure* ***is*** *reduced →* *expenditure* ***are*** *reduced*

Table 4.2: Sample error patterns extracted from NUCLE.

Context window	# PoS patterns	# lexical patterns
0	1,386	5,700
1	6,402	14,568
2	12,623	15,640

Table 4.3: Number of patterns by type extracted from NUCLE.

error patterns in the opposite direction (i.e. *correct → incorrect*). Each of these patterns can also include its frequency in the reference corpus, which is useful if we want to replicate errors keeping their original proportions. For random generation, we adopt a uniform distribution (where all error patterns are considered equally likely) so frequency information is ignored. Two types of patterns are extracted from NUCLE: lexical patterns (in terms of surface forms) and PoS patterns (in terms of PoS tags). Table 4.2 shows some examples. The total number of patterns extracted from NUCLE is presented in Table 4.3.

For each correct sentence in our base texts (EVP), we generate a pseudo-source sentence by applying zero or more error patterns. All the decisions in this process are uniformly random. We first decide whether lexical patterns will be applied, in which case all the relevant error patterns (from the longest to shortest context) are run through the sentence and applied zero or more times with equal probability. This process is repeated with PoS patterns on the same sentence. Thus, lexical patterns and longer contexts take precedence over PoS patterns and shorter contexts, in order to ensure that we use the most useful information first. A sentence can then be distorted by zero or more patterns of different type and context size. The injection of errors is incremental and non-overlapping. Pseudocode for the random generation process is given in Listing 4.1 and a graphical example in Figure 4.1.

A parallel corpus is built using the pseudo-source sentences and the original (correct) versions as target sentences. Since we know what tokens have been deliberately changed to produce the artificial errors, we can also output the corresponding error annotations, although this is not necessary for SMT. Error statistics for each dataset are shown in Table 4.4. Despite being generated with a random method, the EVP dataset shows a fairly accurate error distribution although

```
for each sentence in base-texts do
    pseudosource := sentence;
    target := sentence;
    for each type in (lex, pos) do
        if random(0, 1) > 0.5 then
            for context := 2 to 0 step -1 do
                for each pattern in patterns[type][context] do
                    while pseudosource.matches(pattern.correct) do
                        if random(0, 1) > 0.5 then
                            pseudosource.replace(pattern.correct,
                                                 pattern.incorrect);
    yield pseudosource, target;
```

Listing 4.1: Pseudocode for random AEG.

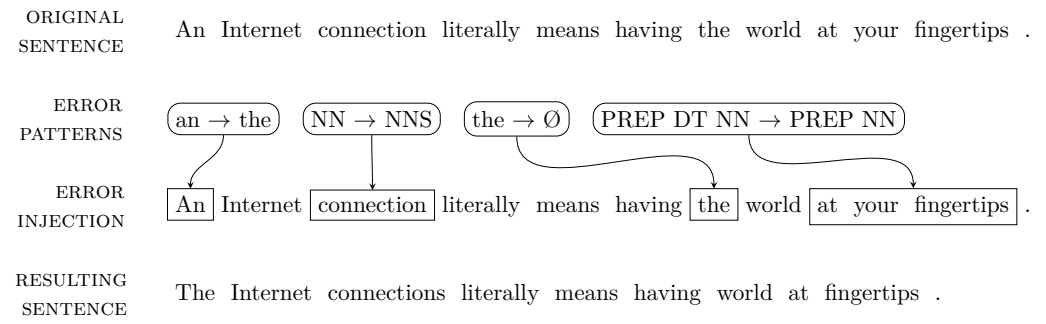


Figure 4.1: An example of the artificial error injection process.

it does exhibit a considerably higher sentence error rate (i.e. the percentage of incorrect sentences in the corpus).

The resulting artificial EVP corpus was used to train SMT systems both in isolation and in combination with NUCLE. Word alignments were generated with Giza++ (Och and Ney, 2003), which implements IBM Models 1-5 and the HMM Model. The final word alignments are built from the intersection of bidirectional runs of Giza++ (from source to target and target to source) plus some additional alignment points from the union of both runs. LMs for the target language were built from the corrected sentences in our parallel corpus using up to 5-grams. For this purpose, we used the IRSTLM Toolkit (Federico et al., 2008) with modified Kneser-Ney smoothing, a refinement of Kneser and Ney's algorithm that uses three different discounting values depending on the original sequence count (Chen and Goodman, 1998).

All our systems were built using the Moses decoder (Koehn et al., 2007), which uses a translation table that associates each phrase pair with five different scores:

- direct and inverse phrase translation probabilities ($\phi(C|E)$ and $\phi(E|C)$),
- direct and inverse lexical weighting ($lex(C|E)$ and $lex(E|C)$), an estimation of the reliability of the phrase translation based on its component lexical probabilities, and

Dataset	Sentence error rate	Error type distribution				
		ArtOrDet	Nn	Prep	SVA	Vform
NUCLE	19.75%	42.08%	23.89%	15.19%	9.65%	9.18%
EVP	84.93%	40.39%	23.82%	16.36%	7.97%	11.45%
CoNLL-2013 test	60.68%	42.00%	24.10%	18.93%	7.55%	7.43%

Table 4.4: Error statistics for each dataset.

- phrase penalty, a constant ($\rho = 2.718$ by default) that controls the preference for fewer (longer) phrases ($\rho < 1$) or more (shorter) phrases ($\rho > 1$).

Default parameters were used for training in all cases. For word alignment, this includes 5 iterations of IBM Model 1, 0 of Model 2, 5 of Model 3, 3 of Model 4, 0 of Model 5 and 5 of the HMM Model. For decoding, the default weights defined in Moses were used (and no tuning was performed). Sentence segmentation, tokenisation and PoS tagging were carried out using NLTK (Bird et al., 2009) for consistency with the CoNLL shared task data, which means that the extracted PoS patterns use the Penn Treebank tagset[2] (see Appendix D). The generation of new word forms that becomes necessary when we use PoS patterns (e.g. VBZ → VBP) was automated with the NodeBox English Linguistics library for Python.[3]

Systems were evaluated on the CoNLL-2013 test set using the M^2 Scorer, the official evaluation tool for the task (see Section 3.1.3). Results are shown in Table 4.5. Following the criteria adopted in the shared task, we measured performance in terms of F_1 but also computed $F_{0.5}$, GLEU and I-measure for comparison. We only report I-measure scores for correction, since this is the only aspect reported by the other metrics.

Systems using artificial data are compared with the original uncorrected text and a baseline system trained on the official NUCLE corpus. Although we are ultimately interested in improving the original text, our immediate goal is to contrast the effects of artificial data with genuine learner data, hence our chosen baseline.

As mentioned in Section 3.1.3, any uncorrected text yields TP = 0 and FP = 0, which produces an undefined value of P due to a division by zero. By convention, P is considered to be 1 (i.e. 100%) in this circumstance but given that this can be misleading for our purpose, we use a dash instead. F_1 and $F_{0.5}$ are also consequently marked, although originally the M^2 Scorer will report 0 for both metrics.

Results in Table 4.5 show that training on random errors alone (EVP) produces a small improvement in R but a significant drop in P with regard to the baseline (NUCLE). The first can be explained by the impact of general PoS patterns, which allow the creation of unseen errors by generating new word forms. This is a desirable aspect of AEG which turned out to be very effective, considering that the EVP corpus is only about a third of NUCLE. The drop in P is a side effect of this strategy,

[2] http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[3] https://www.nodebox.net/code/index.php/Linguistics

Dataset	Original test set						Alternative test set					
	P	**R**	**F_1**	**$F_{0.5}$**	**GLEU**	**I**	**P**	**R**	**F_1**	**$F_{0.5}$**	**GLEU**	**I**
Original text	—	0.00	—	—	78.12	0.00	—	0.00	—	—	87.75	0.00
NUCLE	29.84	8.83	13.62	20.21	77.67	−1.92	40.24	12.55	19.13	27.92	86.77	−1.35
EVP	13.48	**9.74**	11.31	12.52	72.94	−6.63	17.20	**13.06**	14.85	16.17	80.96	−6.14
NUCLE+EVP	24.89	**10.29**	**14.56**	19.39	76.83	−2.96	35.03	**15.15***	**21.15**	27.75	85.82	−2.21

Table 4.5: Results of the NUCLE and EVP datasets on the CoNLL-2013 test set (*original* and *alternative*) using *phrase-based* SMT. Improvements over the baseline (NUCLE) are marked in bold; statistically significant differences are marked with an asterisk.

as the new word forms can also interfere with correct usage and introduce noise. For example, the pattern NN → NNS (which transforms a singular noun into plural), can effectively produce an artificial error such as *modesty* → *modesties* (so that *modesties* is learnt as incorrect) but it can also generate *house* → *houses*, where the second word form is not necessarily incorrect in the original context (e.g. when preceded by the definite article, *the*). In many cases, TPs can also occur when the system flags an infrequent but grammatical phrase as incorrect, often due to a very limited LM. Although we would expect these kinds of phrases at higher proficiency levels, a closer inspection of the results reveals that this happens across all levels and also for common words and phrases, especially if they are particularly frequent in the training data (e.g. *realize* → *realise*, *social media* → *the media*, etc.). As we demonstrated in our submission to the CoNLL-2014 shared task, a large LM (especially one trained on web data) can help lower TPs and increase P (Felice et al., 2014).

The combination of NUCLE+EVP yields even higher R while ameliorating the decrease in P, which is still below the baseline. This combination, however, achieves the highest F_1, which is why we adopted it for our submission to the shared task (Yuan and Felice, 2013). As expected, $F_{0.5}$ is lower than the baseline for systems trained on EVP, given the considerable drop in P. The same occurs for GLEU and I-measure, which are also highly dependent on P. Performance on the alternative test set is consistent with the original test set but naturally higher. Statistical tests (as described in Section 3.4) reveal that only R for NUCLE+EVP is statistically better than for NUCLE ($p < .05$) while all other improvements are not statistically significant.

Following our training strategy for the shared task, we also built SMT systems using PoS-factored models. PoS tags for our data were obtained with the Robust Accurate Statistical Parser (RASP) (Briscoe et al., 2006), which uses the richer CLAWS2 tagset[4] (see Appendix E). Results of these experiments are reported in Table 4.6.

The performance of PoS-factored models is similar to that of phrase-based SMT, exhibiting the same relationship between the datasets. However, it is lower for most of the metrics, with the exception of R and F_1. Since these systems use the probabilities of PoS sequences as extra information to score translations, they can produce more general (but less precise) corrections, thus improving R but harming P.

[4] `http://ucrel.lancs.ac.uk/claws2tags.html`

Dataset	Original test set						Alternative test set					
	P	**R**	**F_1**	**$F_{0.5}$**	**GLEU**	**I**	**P**	**R**	**F_1**	**$F_{0.5}$**	**GLEU**	**I**
Original text	—	0.00	—	—	78.12	0.00	—	0.00	—	—	87.75	0.00
NUCLE	26.08	11.02	15.49	20.48	77.14	−3.08	35.72	15.72	21.83	28.48	85.99	−2.36
EVP	11.78	**12.16**	11.97	11.86	72.44	−7.56	15.27	**16.36**	15.80	15.48	80.23	−6.84
NUCLE+EVP	21.76	**12.84**	**16.15**	19.10	76.30	−3.82	31.10	**18.98***	**23.57**	27.58	85.05	−2.82

Table 4.6: Results of the NUCLE and EVP datasets on the CoNLL-2013 test set (*original* and *alternative*) using *PoS-factored* SMT. Improvements over the baseline (NUCLE) are marked in bold; statistically significant differences are marked with an asterisk.

For example, given the high probability of the pattern VVG RR, we can successfully correct cases such as *eating healthy* → *eating healthily* but we are also more likely to generate false positives for phrases like *feeling good*. However, from the perspective of the CoNLL-2013 shared task, whose goal was to maximise F_1, the hybrid of NUCLE+EVP using a PoS-factored model seems the most advantageous setting.

4.2.1 Error type analysis

Analysing performance by error type is very valuable for system development and tuning. In order to compute traditional metrics for each error type, we must first obtain their corresponding TP, FP and FN counts. However, because only TPs and FNs can be obtained from the gold standard (using the number of matched and missed edits respectively), we are only able to compute R. To compute P, we need to estimate the number of FPs so we first need to classify each superfluous edit proposed by the system into one of the available error types. To do so, we use a set of heuristics that analyse differences in surface forms and PoS tags between the original phrase and the proposed correction in order to infer a type, similar to our strategy in the CoNLL-2013 shared task (Ng et al., 2013). The implemented method has an estimation accuracy of 75.23% on the training set and 87.04% on the test set, which we deemed acceptable for our purposes. Given the difficulty in discriminating between SVA and Vform errors automatically, we merged both types into a single category. Corrections proposed by the system that did not fit into any of the five types were excluded from our analysis.

We report two groups of results. Table 4.7 and Table 4.8 show the performance of the phrase-based system while Table 4.9 and Table 4.10 show results for the PoS-factored system. In line with general results, P is consistently below the baseline for all systems and error types, except Nn in Table 4.10. On the contrary, R improves in many cases, most notably for Nn, SVA and Vform. This reveals that random AEG has a clearly distinct impact on errors involving closed-class words (ArtOrDet and Prep) and open-class words (Nn, SVA and Vform). In the first case, the number of errors that can be generated is always limited, even if PoS patterns are applied. However, in the second case, PoS patterns can generate unseen errors and thus extend coverage in the test set. In many cases, this increase in R can boost F_1, especially

Dataset	ArtOrDet			Nn			Prep			SVA/Vform		
	P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
NUCLE	30.67	13.89	19.12	23.58	6.35	10.01	50.00	3.22	6.05	31.11	5.67	9.59
EVP	11.04	7.38	8.85	18.01	**19.29**	**18.63**	9.72	2.25	3.65	11.50	**10.53**	**10.99**
NUCLE+EVP	26.04	13.60	17.87	22.92	**11.17**	**15.02**	39.39	**4.18**	**7.56**	20.00	**7.29**	**10.68**

Table 4.7: Error type analysis of the *phrase-based* SMT system on the *original* CoNLL-2013 test set. Results in bold show improvements over the baseline (NUCLE).

Dataset	ArtOrDet			Nn			Prep			SVA/Vform		
	P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
NUCLE	39.75	18.58	25.32	36.27	9.20	14.68	55.00	4.21	7.82	47.83	9.73	16.18
EVP	14.89	10.31	12.18	21.80	**22.28**	**22.04**	9.72	2.69	4.21	16.23	**16.44**	**16.34**
NUCLE+EVP	35.48	**19.10**	24.83	35.26	**16.22**	**22.22**	42.42	**5.36**	**9.52**	31.46	**12.39**	**17.78**

Table 4.8: Error type analysis of the *phrase-based* SMT system on the *alternative* CoNLL-2013 test set. Results in bold show improvements over the baseline (NUCLE).

Dataset	ArtOrDet			Nn			Prep			SVA/Vform		
	P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
NUCLE	31.01	15.48	20.65	25.17	9.39	13.68	14.12	3.86	6.06	24.75	10.12	14.37
EVP	11.16	8.22	9.47	19.22	**28.52**	**22.97**	2.74	2.70	2.72	9.15	**18.80**	12.31
NUCLE+EVP	26.33	15.16	19.24	24.47	**16.52**	**19.72**	11.12	**5.01**	**6.91**	15.91	**13.01**	14.32

Table 4.9: Error type analysis of the *PoS-factored* SMT system on the *original* CoNLL-2013 test set. Results in bold show improvements over the baseline (NUCLE).

Dataset	ArtOrDet			Nn			Prep			SVA/Vform		
	P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
NUCLE	40.06	20.23	26.88	35.37	12.84	18.84	21.18	6.82	10.32	39.22	17.54	24.24
EVP	15.06	11.49	13.03	23.27	**32.95**	**27.28**	2.74	3.22	2.97	12.91	**29.37**	17.94
NUCLE+EVP	35.87	**21.29**	26.72	**37.64**	**23.99**	**29.30**	11.98	6.43	8.36	25.03	**22.12**	23.49

Table 4.10: Error type analysis of the *PoS-factored* SMT system on the *alternative* CoNLL-2013 test set. Results in bold show improvements over the baseline (NUCLE).

for Nn. Ranking by F_1 places ArtOrDet first and Prep last, with the other types varying in the middle. This result is directly related to the size of their confusion sets and their function in the language, which is much more constrained and therefore easier to model for articles and determiners. As expected, performance is consistently better on the alternative test set, as it includes complementary annotations.

In summary, we observe that uniformly random generation can produce a small statistically significant improvement in R which can translate into higher F_1, although it entails a considerable drop in P. These results seem to confirm the findings by Sjöbergh and Knutsson (2005) and provide initial hints that artificial data is complementary to error-annotated learner data, as suggested by previous work. The EVP dataset was also used in our submission to the CoNLL-2014 shared task, where it was combined with other training data (Felice et al., 2014). For this submission, however, we used a phrase-based SMT model, as it proved better in terms of P and was in line with the choice of $F_{0.5}$ as the official evaluation metric.

4.3 Probabilistic generation

As described in Section 2.2.3.2, probabilistic approaches to AEG aim to preserve the error distributions observed in the reference corpus. Provided that test data has a similar error distribution to training data, probabilistic methods should produce better results than uniform random methods, which assign equal probability to all errors.

We test this hypothesis in a second round of experiments using different probabilistic configurations. In addition, we identify and control other variables that can help mimic the errors in our reference corpus more accurately. Some of these variables concern the base texts and include:

Topic: Replicating errors in texts about the same topic as the reference corpus is expected to produce better results than out-of-domain data, as vocabulary and word senses are more likely to be similar. In addition, similar texts are more likely to exhibit suitable contexts for error injection and consequently help the system focus on particularly useful information.

Genre: In cases where no a priori information about topic is available (for example, because the test set is unknown or the system will be used in different scenarios), knowing the genre or type of text the system will deal with can also be useful. Example genres include expository (descriptions, essays, reports, etc.), narrative (stories), persuasive (reviews, advertisements, etc.), procedural (instructions, recipes, experiments, etc.) and transactional texts (letters, interviews, etc.).

Style/register: As with the previous variables, style (colloquial, academic, etc.) and register (from formal written to informal spoken) also affect production and should therefore be modelled accurately in the training data.

Text complexity/language proficiency: Candidate texts should exhibit the same reading complexity as the reference corpus and be written by or targeted at learners with similar English proficiency. Otherwise, the overlap in vocabulary and grammatical structures is likely to be small and thus hinder error injection.

Native language: Because L2 production is known to be affected by a learner's L1, using candidate texts produced by groups with the same L1 as the reference corpus should provide more suitable contexts for error injection. When such texts are not available, using data by speakers of other L1s that exhibit similar phenomena (e.g. no article system, Latin languages, etc.) might also be useful. However, finding error-free texts written in English by a specific population can be difficult, which is why most approaches resort to native English text.

These variables are manually controlled in our experiments, although many of them could be assessed automatically. The size of the generated datasets is also an important factor, as the use of more training data normally improves performance.

However, collecting a large amount of learner data to be used as base texts for AEG does not seem very wise, not only because it is more useful as training data on its own but also because we are searching for a way to overcome the low availability of such data by using the vast amounts of available native text instead. Initially, and to comply with the CoNLL shared task requirements to use only publicly available data, we resorted to Wikipedia as a source of base texts.

To control for topic, we chose an initial set of 50 Wikipedia articles based on keywords in the NUCLE training data and proceeded to collect related articles by following hyperlinks in their 'See also' section. We retrieved a total of 494 articles which were later preprocessed to remove wikicode tags, yielding 54,945 sentences and approximately 1,123,739 tokens (closer in size to the NUCLE corpus than our previous EVP dataset). NUCLE and Wikipedia texts have the same genre, style and register (expository, written, academic and formal), although this does not make them necessarily similar. As regards text complexity and language proficiency, essays in the NUCLE corpus are written by advanced university students and are therefore comparable to standard English Wikipedia articles. For a reference corpus with less sophisticated language, the Simple English Wikipedia could be used instead (as we shall see in Chapter 5). Finally, native language seems to be the only discordant variable, since Wikipedia articles are mostly written by native speakers (unlike essays in NUCLE). However, this is exactly the scenario we need to test the usefulness of native data for AEG.

4.3.1 Experimental set-up

Our approach to probabilistic AEG is similar to the one proposed by Rozovskaya and Roth (2010c) in that we use error inflation ($2x$) to boost the naturally low error probabilities (see Listing 4.2). However, unlike them, we refine our probabilities by imposing restrictions on the linguistic functions of the words and the contexts where they occur. Because we extend generation to open-class error types (such as verb form errors), this refinement becomes necessary to overcome disambiguation issues and lead to more accurate replication.

We investigate two main sources of information for generation. The first one is purely statistical and concerns the error type distributions observed in the reference corpus while the second relies on linguistic cues to model the underlying contexts in which errors are likely to occur. We thus generate five distinct datasets based on the following information:

Error type distributions (ED): For each error type, we compute the probability that a relevant instance of the class is an error:

$$p(type) = \frac{errors_{type}}{relevant\ instances_{type}} \tag{4.1}$$

```
inflationFactor := 0.5;
for each sentence in base-texts do
    target := sentence;
    pseudosource := sentence;
    for each type in error-types do
        for each i in relevant-instances(sentence, type) do
            shuffle(alternatives(type));
            for a in alternatives(type) do
                if i = a then /* original = alternative */
                    p := a.probability * inflationFactor;
                else
                    p := a.probability * (1 / inflationFactor);
                x := random(0, 1);
                if x < p then
                    pseudosource.replace(i, a);
                    break;
    yield pseudosource, target;
```

Listing 4.2: Pseudocode for probabilistic AEG with inflation.

During generation, $p(type)$ is uniformly distributed over all the members i in the confusion set for the error type S_{type}, so all of them have the same probability of being chosen as a replacement:

$$p(S^i_{type}|type) = \frac{p(type)}{|S_{type}|} \tag{4.2}$$

For example, if $p(\text{ArtOrDet}) = 0.35$ and the confusion set for English articles is $S_{\text{ArtOrDet}} = \{a, an, the, \emptyset\}$, we have that the probability of changing the article of any given relevant instance to *the* is $p(the|\text{ArtOrDet}) = 0.35/4 = 0.0875$. Relevant instances are detected in the base texts and changed for an alternative at random using the estimated probabilities. The probability of leaving a relevant instance unchanged is $1 - p(type)$.

Morphology (MORPH): We believe that morphological information such as person or number is particularly useful for identifying and correcting article, noun number and subject-verb agreement errors. Thus, we use the probability of a learner producing a given word when another is expected in a specific morphological context, e.g. the probability of producing a specific article given a singular noun that requires the indefinite article *an* (as shown below).

P(source=*an*|target=*an*, head-noun=NN)	= 0.942
P(source=*the*|target=*an*, head-noun=NN)	= 0.034
P(source=*a*|target=*an*, head-noun=NN)	= 0.015
P(source=other|target=*an*, head-noun=NN)	= 0.005
P(source=Ø|target=*an*, head-noun=NN)	= 0.004

PoS disambiguation (POS): Most approaches to AEG are aimed at closed-class words such as articles or prepositions, which rarely occur with a different PoS in the text. However, open-class words can play different roles in a sentence, so PoS disambiguation becomes necessary. As illustrated below, error probabilities for the same source form (e.g. *play*) can vary according to its PoS so only the ones corresponding to the observed PoS are applied.

$$\begin{aligned}
&P(\text{source}=play|\text{target}=play,\ \text{PoS}=V) &&= 0.98\\
&P(\text{source}=plays|\text{target}=play,\ \text{PoS}=V) &&= 0.02\\
&P(\text{source}=play|\text{target}=play,\ \text{PoS}=N) &&= 0.84\\
&P(\text{source}=plays|\text{target}=play,\ \text{PoS}=N) &&= 0.16
\end{aligned}$$

Semantic classes (SC): We hypothesise that semantic information about concepts in the text can reveal hidden usage patterns. Therefore, we use detailed confusion probabilities for articles and prepositions depending on the class of the head noun (e.g. a location, a recipient, an instrument, etc.). For example:

$$\begin{aligned}
&P(\text{source}=in|\text{target}=in,\ \text{noun-class}=\text{location}) &&= 0.39\\
&P(\text{source}=at|\text{target}=in,\ \text{noun-class}=\text{location}) &&= 0.31\\
&P(\text{source}=to|\text{target}=in,\ \text{noun-class}=\text{location}) &&= 0.16\\
&P(\text{source}=from|\text{target}=in,\ \text{noun-class}=\text{location}) &&= 0.07\\
&P(\text{source}=\emptyset|\ \text{target}=in,\ \text{noun-class}=\text{location}) &&= 0.05\\
&P(\text{source}=\text{other}|\text{target}=in,\ \text{noun-class}=\text{location}) &&= 0.03
\end{aligned}$$

By abstracting from surface forms, we can also generate faithful errors for words that have not been previously observed, e.g. we may have not seen *hospital* but we may have seen *school*, *my sister's house* or *church*.

Word senses (WSD): Polysemous words with the same PoS can exhibit different patterns of usage for each of their meanings (e.g. one meaning may co-occur with a specific preposition more often than the others). For this reason, we use probabilities for each word sense in an attempt to capture more accurate usage. As an example, consider a hypothetical situation in which a group of learners confuse prepositions used with the word *bank* as a financial institution (sense 1) but they produce the right preposition when it refers to a river bed (sense 2):

$$\begin{aligned}
&P(\text{source}=in|\text{target}=in,\ \text{head-noun}=bank_1) &&= 0.76\\
&P(\text{source}=at|\text{target}=in,\ \text{head-noun}=bank_1) &&= 0.18\\
&P(\text{source}=\text{other}|\text{target}=in,\ \text{head-noun}=bank_1) &&= 0.06\\
&P(\text{source}=on|\text{target}=on,\ \text{head-noun}=bank_2) &&= 1.00
\end{aligned}$$

Although it is rare that occurrences of the same word will refer to different meanings within a document (the so-called 'one sense per discourse' assumption (Gale et al., 1992)), this is not the case for corpora comprised of different articles. In such scenarios, word sense disambiguation should produce more accurate results.

Information	**Probability**	**Generated error types**
Error type distributions	P(*error_type*)	ArtOrDet, Nn, Prep, SVA, Vform
Morphology	P(source=*determiner*|target=*determiner*, *head_noun_tag*) P(source=*verb_tag*|target=*verb_tag*, *subj_head_noun_tag*)	ArtOrDet, SVA
PoS disambiguation	P(source=*word*|target=*word*, *PoS*)	Nn, Vform
Semantic classes	P(source=*determiner*|target=*determiner*, *head_noun_class*) P(source=*preposition*|target=*preposition*, *head_noun_class*)	ArtOrDet, Prep
Word senses	P(source=*preposition*|*verb_sense* + *obj_head_noun_sense*) P(source=*preposition*|target=*preposition*, *head_noun_sense*) P(source=*preposition*|target=*preposition*, *dep_adj_sense*) P(source=*determiner*|target=*determiner*, *head_noun_sense*) P(source=*verb_tag*|target=*verb_tag*, *subj_head_noun_sense*)	ArtOrDet, Prep, SVA

Table 4.11: Probabilities computed for each type of linguistic information and error types they can generate.

Part of speech	**WordNet classification**
Adjective	all, pertainyms, participial
Adverb	all
Noun	act, animal, artifact, attribute, body, cognition, communication, event, feeling, food, group, location, motive, object, person, phenomenon, plant, possession, process, quantity, relation, shape, state, substance, time
Verb	body, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social, stative, weather

Table 4.12: WordNet classes for content words.

Table 4.11 summarises the probabilities used to generate each dataset and the error types to which they are applicable.

Given the aims of the CoNLL-2013 shared task, we trained our new systems using PoS-factored models, as they achieved the highest F_1 performance in our random generation experiments. PoS tagging was performed with RASP (Briscoe et al., 2006) and word sense disambiguation was carried out using the WordNet::SenseRelate:AllWords Perl module (Pedersen and Kolhatkar, 2009) which assigns a sense from WordNet (Miller, 1995) to each content word in a text. For semantic classes, we used the ones provided by WordNet, which are readily available in NLTK (Bird et al., 2009). WordNet classes respond to a classification in lexicographers' files[5] and are defined for content words as shown in Table 4.12, depending on their location in the hierarchy.

We created one dataset for each type of information in Section 4.3.1. The sets of probabilities defined for each dataset were computed on our reference corpus (NUCLE) and then used to generate artificial errors on the base texts (Wikipedia articles). Error statistics for the resulting datasets are shown in Table 4.13, along with NUCLE and the CoNLL-2013 test set for comparison.

[5] `http://wordnet.princeton.edu/man/lexnames.5WN.html`

Dataset	Sentence error rate	Error type distribution				
		ArtOrDet	Nn	Prep	SVA	Vform
NUCLE	19.75%	42.08%	23.89%	15.19%	9.65%	9.18%
ED	14.58%	43.59%	23.18%	16.47%	8.96%	7.80%
MORPH	13.13%	93.27%	0.00%	0.00%	6.73%	0.00%
POS	13.59%	0.00%	71.34%	0.00%	0.00%	28.66%
SC	43.09%	20.80%	0.00%	79.20%	0.00%	0.00%
WSD	11.45%	71.91%	0.00%	19.34%	8.74%	0.00%
CoNLL-2013 test	60.68%	42.00%	24.10%	18.93%	7.55%	7.43%

Table 4.13: Error statistics for each probabilistic dataset.

Dataset	Original test set						Alternative test set					
	P	R	F_1	$F_{0.5}$	GLEU	I	P	R	F_1	$F_{0.5}$	GLEU	I
Original text	—	0.00	—	—	78.12	0.00	—	0.00	—	—	87.75	0.00
NUCLE	26.08	11.02	15.49	20.48	77.14	−3.08	35.72	15.72	21.83	28.48	85.99	−2.36
ED	**26.11**	3.23	5.74	10.79	**77.80**	**−0.82***	35.47	4.66	8.23	15.27	**87.17***	**−0.61***
MORPH	18.18	4.50	7.21	11.31	76.97	**−1.99***	22.36	5.87	9.30	14.32	**86.06**	**−1.79**
POS	**29.79**	2.56	4.71	9.53	**78.06**	**−0.54***	**43.26**	3.93	7.21	14.41	**87.46***	**−0.33***
SC	12.84	4.87	7.06	9.67	75.83	−3.47	15.41	6.19	8.83	11.87	84.52	−3.28
WSD	21.19	4.99	8.08	12.85	**77.25**	**−1.79***	26.10	6.52	10.43	16.31	**86.27**	**−1.58***
NUCLE+ED	**29.62**	10.53	**15.54**	**21.74**	**77.78**	**−2.25***	**41.37***	15.31	**22.35**	**30.86**	**86.76**	**−1.53***
NUCLE+MORPH	**27.63**	9.92	14.60	20.36	**77.57**	**−2.39***	**38.31**	14.31	20.84	**28.69**	**86.53**	**−1.71***
NUCLE+POS	**31.00**	9.98	15.10	**21.81**	**77.95**	**−1.98***	**43.10***	14.44	21.63	**30.85**	**87.00***	**−1.31***
NUCLE+SC	24.92	9.86	14.13	19.09	**77.20**	**−2.84**	34.82	14.35	20.32	27.09	**86.12**	**−2.15**
NUCLE+WSD	**28.30**	9.92	14.69	**20.65**	**77.61**	**−2.32***	**39.34**	14.37	21.05	**29.19**	**86.62**	**−1.65***

Table 4.14: Results of NUCLE and probabilistic datasets on the CoNLL-2013 test set (*original* and *alternative*) using *PoS-factored* SMT. Improvements over the baseline (NUCLE) are marked in bold; statistically significant differences are marked with an asterisk. Results show that artificial data can increase P but at the expense of R.

Sentence error rates are fairly homogeneous across the datasets, except for SC, which contains roughly 3 times the number of incorrect sentences in the rest of the artificial corpora. Error type distributions are generally dissimilar, with ED being, logically, the most similar to NUCLE. Given the definitions in Table 4.11, ED is the only dataset that contains instances for the five error types, whereas the others only contain instances for a subset of them.

Systems were trained using each dataset in isolation as well as in combination with the NUCLE corpus. Performance on the CoNLL-2013 test set is reported in Table 4.14. Systems using artificial datasets alone are unable to beat the baseline in terms of R, F_1 and $F_{0.5}$ although ED, POS and WSD do show improvements in GLEU and I-measure. For ED and POS, these improvements are statistically significant, especially on the alternative test set. These two datasets also show a slight improvement in P on the original test set but they were found not to be significant. In all cases, a considerable drop in R and F scores is also observed, which suggests that the probabilities used by our methods are perhaps too specific and thus fail to generalise to other words in the data. The SC dataset is clearly the worst performing, indicating that errors in the test set are not really driven by the kind of semantic information encoded by our method.

Hybrid systems trained on NUCLE plus artificial datasets perform better overall and all of them beat the baseline on GLEU and I. Only NUCLE+ED is better than the baseline for the rest of metrics (except R), making it the best of all systems. This is partly expected since the ED dataset is the only one containing instances for all error types and preserving the original error distribution, although it is the least linguistically-informed method. NUCLE+POS and NUCLE+WSD are also among the best performing systems while NUCLE+SC is still the worst.

It could be argued that the reason for these improvements is corpus size, since our hybrid datasets are double the size of individual datasets. However, a comparison of performance between the two groups (e.g. in terms of P, GLEU and I) seems to contradict this hypothesis. The consistent rise in performance when using hybrid sets adds more evidence to our initial results suggesting that artificial data is complementary to genuine learner data.

4.3.2 Error type analysis

Given that the evaluated datasets do not include instances for the same error types, it seems more appropriate to analyse performance by error type. Table 4.15 and Table 4.16 report results by type for the original and alternative sets respectively.

Results show that performance varies across datasets, suggesting that certain types of information are better suited to specific error types. In particular, we find that when purely artificial datasets are used (on both the original and alternative test set), the best F_1 scores per type correspond to WSD for ArtOrDet, POS for Nn and SVA/Vform, and SC for Prep, which is the only combination that beats the baseline. This is mainly the result of an increase in R by the use of semantic classes, which clearly influence the choice of preposition (e.g. *to* + location, *with* + artifact, *during* + time, etc.). By contrast, ED is the worst performing on Prep, as generating preposition errors just based on confusion probabilities does not guarantee that the new choice is an actual error (e.g. *in* → *at*) and therefore causes confusion during training.

Hybrid datasets perform better overall, especially when evaluated on the alternative test set. On this set, there is at least one hybrid dataset that beats the baseline on each error type: NUCLE+ED/MORPH/WSD for ArtOrDet, NUCLE+ED/POS for Nn, NUCLE+SC/WSD for Prep and NUCLE+ED for SVA/Vform. Generating errors based on error type distributions (ED) therefore seems the most versatile method, consistently ranking among the top results for three out the four error categories.

P was found to increase in all cases for ArtOrDet and SVA/Vform, although it never exceeds the minimum value of 50 that would indicate improvement of the source text. The only error type fulfilling this condition is Nn on the alternative dataset, with a maximum P score of 70 using NUCLE+ED. This means that the number of FPs is small in this case, which is not surprising given the high proportion of such errors and the limited size of its confusion set (which is basically binary).

Dataset	ArtOrDet			Nn			Prep			SVA/Vform		
	P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
NUCLE	27.16	15.51	19.74	46.25	9.34	15.54	13.33	3.86	5.99	26.04	10.16	14.62
ED	**28.13**	3.91	6.87	**65.79**	6.31	11.52	2.33	0.32	0.56	0.00	0.00	—
MORPH	18.62	10.58	13.49	—	0.00	—	0.00	0.00	—	14.29	0.41	0.80
POS	0.00	0.00	—	44.05	9.34	15.41	0.00	0.00	—	15.15	2.03	3.58
SC	16.83	7.39	10.27	—	0.00	—	9.86	**9.32**	**9.58**	0.00	0.00	—
WSD	22.19	10.29	14.06	0.00	0.00	—	**19.05**	2.57	4.53	18.75	1.22	2.29
NUCLE+ED	**31.85**	13.48	18.94	**54.65**	**11.87**	**19.50**	13.04	3.86	5.96	**26.58**	8.54	12.93
NUCLE+MORPH	**28.57**	15.07	19.73	45.90	7.07	12.25	**17.19**	3.54	5.87	**28.17**	8.13	12.62
NUCLE+POS	**33.84**	12.90	18.68	**46.59**	**10.35**	**16.94**	**18.84**	**4.18**	**6.84**	**26.25**	8.54	12.89
NUCLE+SC	**28.90**	12.90	17.84	45.00	6.82	11.84	**14.92**	**8.68**	**10.98**	**28.36**	7.72	12.14
NUCLE+WSD	**30.03**	14.49	19.55	**46.67**	7.07	12.28	**19.48**	**4.82**	**7.73**	**26.32**	8.13	12.42

Table 4.15: Error type analysis of our *PoS-factored* SMT systems using probabilistic AEG and tested on the *original* CoNLL-2013 test set. Results in bold show improvements over the baseline (NUCLE).

Dataset	ArtOrDet			Nn			Prep			SVA/Vform		
	P	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**	**P**	**R**	**F_1**
NUCLE	35.19	20.26	25.72	61.63	13.02	21.50	20.69	6.82	10.26	41.05	17.18	24.22
ED	**40.63**	5.79	10.14	**72.97**	6.84	12.51	4.65	0.77	1.32	18.18	1.83	3.33
MORPH	22.70	13.11	16.62	—	0.00	—	0.00	0.00	—	28.57	0.92	1.78
POS	0.00	0.00	—	54.65	11.69	19.26	0.00	0.00	—	**42.42**	6.31	10.99
SC	21.12	9.44	13.05	—	0.00	—	10.88	**12.21**	**11.51**	0.00	0.00	—
WSD	27.81	13.13	17.84	0.00	0.00	—	**21.43**	3.47	5.97	20.00	1.38	2.58
NUCLE+ED	**43.34**	18.49	**25.92**	**70.00**	**15.52**	**25.41**	16.85	5.75	8.57	**47.44**	16.30	**24.26**
NUCLE+MORPH	**37.91**	20.06	**26.24**	**63.08**	10.17	17.52	**22.95**	5.36	8.69	**47.14**	14.54	22.22
NUCLE+POS	**46.01**	17.61	25.47	60.87	**13.83**	**22.54**	**24.24**	6.13	9.79	**44.30**	15.49	22.95
NUCLE+SC	**39.61**	17.73	24.50	61.54	9.93	17.10	18.44	**12.50**	**14.90**	**48.48**	14.10	21.85
NUCLE+WSD	**39.94**	19.33	**26.05**	**63.08**	10.17	17.52	**24.32**	**6.90**	**10.75**	**46.67**	15.35	23.10

Table 4.16: Error type analysis of our *PoS-factored* SMT systems using probabilistic AEG and tested on the *alternative* CoNLL-2013 test set. Results in bold show improvements over the baseline (NUCLE).

Overall, we observe that the error types with highest average F_1 (in decreasing order) are ArtOrDet, Nn, SVA/Vform and Prep. Not coincidentally, this ranking correlates closely with the size of their confusion sets.

4.4 Analysis and discussion

A comparison between random and probabilistic AEG reveals some interesting differences. First of all, their effect on evaluation metrics seems to go in opposite directions; while random generation increases R at the expense of P, probabilistic generation increases P and lowers R. In turn, this translates into improvements in F_1 and $F_{0.5}$ respectively so the choice of one method over the other will eventually depend on the metrics that we seek to maximise. If we choose F_1 (as in the CoNLL-2013 shared task), random generation looks superior whereas for $F_{0.5}$ (as in the CoNLL-2014 shared task), probabilistic methods fare better.

However, it is not clear whether the methods themselves or the differences in the base texts (corrected learner data vs. native text) are responsible for this behaviour. In any case, the purpose of AEG is to provide an inexpensive way to produce error-annotated data from native text, so persisting in the use of learner data seems

impractical and unrealistic. Additionally, given the arguments in favour of P for GEC (see Section 3.1.1), we believe $F_{0.5}$ should be preferred, in which case probabilistic methods should be more advantageous.

Yet, there are considerable variations in performance among our probabilistic datasets. This is partly due to the fact that certain types of information are only applied to specific error types while in other cases it is due to intrinsic limitations of the linguistic information used to characterise the errors. For example, morphological information does not perform as expected on SVA/Vform errors, suggesting that the association between a verb and its head-noun PoS tags is not enough to discriminate between correct and incorrect forms. In other cases, linguistic information is clearly beneficial, showing that errors are indeed influenced by the encoded features. Some examples include SC for Prep and POS for Nn.

The least linguistically-motivated dataset, ED, seems the most robust, possibly because it does not assume any underlying linguistic conditions and relies solely on frequency, so it generalises better. Although linguistic information can be useful for specific error types, only POS is comparable to ED in terms of generalisation power. This suggests that using sophisticated linguistic processing for AEG is not as useful as initially expected and is probably not worth pursuing further. Likewise, the use of error inflation in our methods was unable to boost R as expected, in contrast to the results by Rozovskaya and Roth (2010c), although this might be due to differences in the training paradigm.

Finally, although some systems are able to beat the baseline trained on NUCLE alone, none of them improves the source text in reality, since values of P are below 50, GLEU is always lower than for the original text, and I-measure scores are negative in all cases.

4.4.1 Comparison with systems in the CoNLL-2013 shared task

We carried out a comparison of our models with the 17 participating systems in the CoNLL-2013 shared task, showing how they would rank in terms of F_1 (see Table 4.17). As expected, the system trained on NUCLE+EVP achieves the best result (which is similar to our submission to the task) with the one trained on NUCLE+ED as a runner-up. Of all the systems, these two are the only ones that outperform the baseline trained on NUCLE alone and are ranked above average.

This suggests that using an off-the-shelf SMT system trained on a combination of real and artificial data can yield better results than other machine learning techniques (Berend et al., 2013; Bosch and Berck, 2013; Yi et al., 2013) or rule-based approaches (Flickinger and Yu, 2013; Kunchukuttan et al., 2013; Putra and Szabo, 2013; Sidorov et al., 2013). The implications of these results are very encouraging, since they prove that using the simplest artificial datasets as additional training data can produce competitive results, if not beat highly-engineered systems.

Dataset	Original test set		Alternative test set	
	F_1	**Rank**	**F_1**	**Rank**
NUCLE	15.49	9	21.83	8
EVP	11.97	9	15.80	9
NUCLE+EVP	**16.15**	**8**	**23.57**	**7**
ED	5.74	14	8.23	14
MORPH	7.21	12	9.30	12
POS	4.71	14	7.21	14
SC	7.06	12	8.83	12
WSD	8.08	10	10.43	11
NUCLE+ED	**15.54**	9	**22.35**	**7**
NUCLE+MORPH	14.60	9	20.84	9
NUCLE+POS	15.10	9	21.63	8
NUCLE+SC	14.13	9	20.32	9
NUCLE+WSD	14.69	9	21.05	9

Table 4.17: Rankings of our systems using random and probabilistic AEG with respect to the 17 participating systems in the CoNLL-2013 shared task. Results in bold show improvements over the baseline (NUCLE).

Chapter 5

Experiments on general error correction

In this chapter, we draw on findings from previous experiments to refine our probabilistic AEG method and generate more accurate datasets for general error correction. Since we now focus on a much larger set of errors, we use the CLC as our reference corpus and control more variables during generation. In the first section, we describe our experimental set-up, including the data and generation method. In the second section, we present and discuss the results of our experiments in order to discover the best-performing settings.

5.1 Experimental set-up

In the light of results in Chapter 4 and following our guiding principles, we conduct experiments on general error correction using only the best performing settings from previous experiments. Given the current trend towards emphasising the role of P in the evaluation of GEC systems, we focus only on methods that maximise this metric. Thus, the experiments presented in this chapter rely on probabilistic methods using information from error distributions and PoS patterns, as they have proved the most successful in this regard. Similarly, we only train phrase-based models, as they maximise P and related metrics such as $F_{0.5}$, GLEU and I. This has the added benefit of giving us 'purer' results, without the effect of additional factors.

5.1.1 Data

For these new experiments on general error correction, we adopt the CLC as our working corpus (see Section 2.2.2.1), as it has higher annotation quality than NUCLE and is more representative of L2 writing given its variety of learner backgrounds, proficiency levels, error categories and size.

Dataset	# scripts	# sentences	# tokens
CLC-train	131,777	1,965,727	28,823,615
FCE-test	97	2,691	41,464

Table 5.1: Datasets used in our probabilistic generation experiments for general error correction.

We split the CLC into the following two subsets:

CLC-train: A portion of the error-coded version of the CLC, containing 131,777 scripts from all the examinations represented in the corpus. This is used as our reference corpus for collecting all the necessary statistics for probabilistic AEG.

FCE-test: A subset of 97 scripts from the FCE public dataset, as defined by Yannakoudakis et al. (2011). These texts have been written by learners mostly at the B2 (upper intermediate) level in the CEFR, which we deem suitable for testing purposes.

The composition of these datasets is shown in Table 5.1.

For base texts, we use two different collections of articles from English Wikipedia and Simple English Wikipedia respectively, in order to study the effect of language complexity on AEG. The artificial datasets created from this data are described in Section 5.1.4. Text preprocessing was carried out with RASP.

5.1.2 Error patterns

We generate artificial errors on error-free text using *correct* → *incorrect* patterns extracted from the reference corpus, as in Section 4.2. However, there are a number of differences in the way they are extracted and applied. First, we create different sets of patterns for each maximum context window size (0, 1 and 2) and apply them separately in order to study how this affects the quality of the generated errors.

Second, we redefine the way PoS patterns are built, so that modified tokens are represented by their surface forms and PoS tags, as illustrated in Table 5.2. This allows us to replace word forms without recourse to external libraries, making it easier to extend generation to all error types and, potentially, to other languages too. Although we lose the ability to generate new word forms, AEG should be more accurate, as we limit the set of candidate words to the ones that learners really find confusing and we also minimise the chance of constructing artificial errors that result in correct usage. In other words, we trade errors on new words for well-known errors in new contexts, as is also done by Wagner (2012).

Finally, we associate each pattern with the error types and frequencies observed in the reference corpus, which allows us to retrieve patterns by type and have control over the generated distribution. For example, the pattern */VV0 *a*/AT1 *lot*/NN1 *of*/IO */NN2 → */VV0 *many*/DA2 */NN2 has been tagged in three different ways

PoS pattern	**Example**
cannot/VM → can/VM not/XX	*I **cannot** go → I **can not** go*
*/VM */RR */VB0 */JJ → */VM */VB0 */RR */JJ	*We will **never be** able to afford it → We will **be never** able to afford it*
*/RL */TO hearing/VVG */II */PPY → */RL */TO hear/VV0 */II */PPY	*I look forward to **hearing** from you → I look forward to **hear** from you*

Table 5.2: Sample PoS patterns for probabilistic AEG.

Frequency threshold	**# PoS patterns**	**# lexical patterns**
1	2,133,312	2,604,023
2	206,999	97,573
3	92,733	36,458
4	56,758	20,540
5	39,729	13,845
6	29,857	10,114
7	23,693	7,767
8	19,476	6,256
9	16,290	5,217
10	14,003	4,430

Table 5.3: Number of patterns by type extracted from CLC-train, corresponding to a maximum context window size of 2.

with varying frequency: twice as a wrong quantifier because of noun countability (CQ), 43 times as a quantifier replacement (RQ) and 261 times as inappropriate register (L).

To further minimise the chance of generating errors that may be grammatically correct and avoid rare unrepresentative errors, we set a frequency threshold for patterns, so that only those with frequency greater than or equal to the threshold are used during generation. For our experiments, this threshold is set to 10. There is obviously a trade-off when choosing this value: while a low frequency threshold will expand the spectrum of errors we can generate, it can also render the generation process slow and impractical. For a large dataset such as the training portion of the CLC, the number of error patterns increases quickly for lower frequency thresholds, as shown in Table 5.3.

One distinctive aspect of the CLC is the annotation of nested errors, i.e. a sequence of embedded errors that are corrected one by one. Figure 5.1 illustrates one such case. In this example, the noun *relation* should be changed to the verb *relate*, which in turn should agree with the subject of the sentence, thus *relates*. As it is difficult to decompose each nested error into meaningful individual errors for AEG, we use the outermost corrections and treat them as atomic units, whose type is composed of all inner types. For the example above, we would extract *relation* → *relates*, rewrite it as *relates* → *relation* and classify it as AGV+DV. As expected, this expands the number of error types and creates a new distribution, which in the case of CLC-train amounts to 5,302 composite types. The first 100 most frequent are listed in Appendix F.

```
Almost everyone <NS type="AGV"><i><NS type="DV"><i>relation</i>
<c>relate</c></NS></i><c>relates</c></NS> famous people with
photographers, articles, money, fans, ...
```

Figure 5.1: A nested error from a public portion of the CLC.

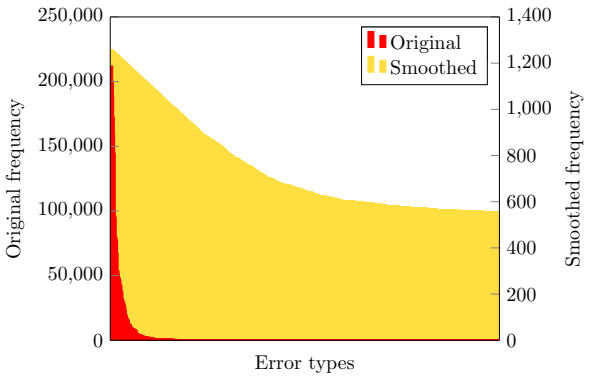


Figure 5.2: Effect of smoothing on the distribution of error types in CLC-train.

5.1.3 Generation method

The approach to probabilistic generation adopted in these new experiments imposes more control on the statistical aspects of the generated datasets, including the proportion of incorrect sentences (i.e. the sentence error rate), the average number of errors per sentence, the distribution of error types and the probability of each error pattern within each type. The first two variables are preserved as in the original reference corpus while the other two depend on the generation mode.

We implemented three different generation modes to control the distribution of error types and patterns:

Original: Uses the original probability distribution observed in the reference corpus.

Uniform: Uses a uniform probability distribution, so all elements are equally likely.

Smoothed: Redistributes the probability mass among all elements in order to minimise extreme differences while maintaining the relationship between them. This is similar to smoothing techniques used in language modelling. Listing 5.1 shows our implemented algorithm while Figure 5.2 illustrates the effect of smoothing on the distribution of error types in CLC-train. Using this technique, we seek to transform the original Zipfian distribution in order to boost the probability of under-represented errors.

Listing 5.2 shows the pseudocode for our refined probabilistic generation method. For each sentence in our base texts, we first determine the number of errors that we will create, using weighted random selection to preserve the original distribution. We then attempt to generate each required error by selecting a type and an applicable

```
function smooth(list_of_frequencies):
    f := list_of_frequencies;
    c := [];
    for key, group in group_by(list_of_frequencies) do
        n := len(list(group));
        c.extend([n] * n);
    changed := True;
    while changed do
        changed := False;
        i := len(f) - 1;
        while i > 0 do
            i := i - c[i] + 1;
            j := i;
            new_f := ((f[i] * c[i]) - c[j-1]) / c[i];
            while new_f > f[i-1] and new_f > f[j-1] + 1 do
                for k := j - c[j-1] to j-1 do
                    f[k] := f[k] + 1;
                for k := i to i + c[i] - 1 do
                    f[k] := new_f;
                changed := True;
                j := j - c[j-1];
                if j > 0 then
                    new_f := ((f[i] * c[i]) - c[j-1]) / c[i];
                else
                    break;
                i = i - 1;
    return f;
```

Listing 5.1: The implemented smoothing algorithm.

pattern, according to the probability distribution dictated by the chosen generation mode. If it is not possible to generate an error for the chosen type, a random error is attempted for any one of the remaining types. Sometimes it is just impossible to generate the required number of errors in a given sentence, either because the sentence is exhausted or none of the error patterns is applicable.

The method also outputs pairs of error-free sentences in order to keep the required proportion of correct and incorrect sentences in the generated dataset. Finally, an adaptive mechanism controls the number of artificial errors per sentence so as to keep the average as close as possible to the original.

5.1.4 Generated datasets

We generated artificial datasets using different combinations of our independent variables, namely:

- base texts (English Wikipedia or Simple English Wikipedia),
- maximum context window (0, 1 or 2),
- pattern type (PoS or lexical), and
- generation mode (original, uniform or smoothed).

```
/* mode is ORIGINAL, UNIFORM or SMOOTHED */
num_errors := get_num_errors_distribution(reference-corpus);
error_types := get_error_type_distribution(reference-corpus, mode);
candidate_num_errors := num_errors;
avg_e := average(num_errors);
p_correct := num_errors[0] / sum(num_errors)
type_counts = {}
n_correct := 0
ge_total := 0;
s := 0;
for each sentence in base-texts do
    target := sentence;
    pseudosource := sentence;
    e := get_random(candidate_num_errors, ORIGINAL);
    ge := 0;
    if e > 0 then
        forced_random := OFF;
        candidate_types := error_types;
        while (e - ge) > 0 or forced_random = ON do
            generated := False;
            while not generated and len(candidate_types) > 0 do
                if forced_random = ON then
                    t := get_random(candidate_types, UNIFORM);
                    candidate_types := error_types;
                    forced_random := DONE;
                else
                    found := False;
                    for t in candidate_types do
                        if ge_total = 0 or
                           type_counts[t] / ge_total < t.probability then
                            found := True;
                            break;
                    if not found then
                        forced_random := ON;
                        break;
                candidate_patterns := get_patterns(t, mode);
                while not generated and len(candidate_patterns) > 0 do
                    p := get_random(candidate_patterns, mode);
                    matches := get_matches(p, pseudosource);
                    if matches then
                        m := get_random(matches, UNIFORM);
                        pseudosource.replace(m, p.incorrect);
                        generated := True;
                        ge := ge + 1;
                        type_counts[t] := type_counts[t] + 1;
                    else
                        candidate_patterns.remove(p);
                if not generated then
                    candidate_types.remove(t);
            if not generated and forced_random = DONE then
                break;
        if ge > 0 then
            s := s + 1;
            ge_total := ge_total + ge;
            yield pseudosource, target
    if n_correct <= prob_correct * s then
        s := s + 1;
        n_correct = n_correct + 1;
        yield target, target;
    new_avg_e := ge_total / (s - n_correct);
    if new_avg_e < avg_e then
        candidate_num_errors := filter_out(num_errors <= avg_e);
    else if new_avg_e > avg_e then
        candidate_num_errors := filter_out(num_errors >= avg_e);
```

Listing 5.2: Pseudocode for our refined probabilistic AEG.

The number of possible datasets that can be generated from these variables amounts to $2 \times 3 \times 2 \times 3 = 36$. However, because we are mostly interested in generalising to other errors, we conduct our experiments using PoS patterns and only test lexical patterns in selected cases for comparison. As explained in Section 5.1.2, the minimum frequency threshold was set to 10.

Each artificial dataset created contains 1,965,727 sentences, matching the size of CLC-train. For simplicity, we will refer to each dataset using a unique identifier based on its composition, as shown below:

[EW|SW]-[0|1|2]-[POS|LEX]-[O|U|S]

For example, EW-0-POS-O refers to an artificial dataset using English Wikipedia, 0 context window, PoS patterns and the original error type distribution.

Since we now work with a large number of error types (both single and combined), it would be impractical to report their individual proportions. Instead, we compare the error type distributions between CLC-train and each dataset directly by computing the Jensen-Shannon Divergence (JSD) (Lin, 1991). This popular method for computing similarity between probability distributions is based on the Kullback–Leibler Divergence (KLD) (Kullback and Leibler, 1951) but has a number of advantages, such as being symmetric and giving values within finite intervals. JSD for two discrete distributions, P and Q, is defined in Equation 5.1 and gives values in the $[0, 1]$ interval. When $P = Q$, $JSD(P \parallel Q) = 0$.

$$JSD(P \parallel Q) = \frac{1}{2} KLD\left(P \,\middle\|\, \frac{P+Q}{2}\right) + \frac{1}{2} KLD\left(Q \,\middle\|\, \frac{P+Q}{2}\right) \tag{5.1}$$

where

$$KLD(P \parallel Q) = \sum_i P(i) \log_2 \frac{P(i)}{Q(i)} \tag{5.2}$$

Statistics for our PoS-based probabilistic datasets are given in Table 5.4. Artificial datasets with lower JSD are more similar to CLC-train and thus expected to convey roughly the same error information. As expected, JSD is highest when using uniform distributions and lowest when preserving the original. The use of a larger context window also seems to help, except for the original distribution where JSD increases. We also observe that smaller contexts tend to generate more errors per sentence as there are more possible insertion points, and using different versions of Wikipedia does not seem to affect similarity in principle. Sample sentences for different combinations of our generation variables are included in Appendix G.

5.2 Experiments and results

We used the generated datasets to build phrase-based SMT systems using Giza++ and Moses with default parameters. A baseline system was trained on CLC-train

Dataset	Sentence error rate	Average errors per sentence	JSD
CLC-train	61.41%	2.46	—
EW-0-POS-O	61.41%	1.52	0.0061
EW-0-POS-U	61.41%	1.19	0.4441
EW-0-POS-S	61.41%	1.12	0.4409
EW-1-POS-O	61.41%	1.33	0.0197
EW-1-POS-U	61.41%	1.01	0.3080
EW-1-POS-S	61.41%	1.01	0.3026
EW-2-POS-O	61.41%	1.07	0.0232
EW-2-POS-U	61.41%	1.00	0.2303
EW-2-POS-S	61.41%	1.01	0.2294
SW-0-POS-O	61.41%	2.44	0.0060
SW-0-POS-U	61.41%	1.34	0.4553
SW-0-POS-S	61.41%	1.28	0.4504
SW-1-POS-O	61.41%	1.36	0.0170
SW-1-POS-U	61.41%	1.04	0.3212
SW-1-POS-S	61.41%	1.03	0.3170
SW-2-POS-O	61.41%	1.06	0.0238
SW-2-POS-U	61.41%	1.00	0.2194
SW-2-POS-S	61.41%	1.00	0.2178

Table 5.4: Error statistics for PoS-based probabilistic datasets based on CLC-train.

while the rest of systems were trained on artificial datasets, both in isolation and in combination with the former corpus. Performance of these systems on FCE-test are reported in Table 5.5.

Results show that artificial datasets used in isolation are not able to outperform the CLC-train baseline, except for EW-1-POS-U and EW-2-POS-O which achieve a higher I-measure score (although it is not statistically significant). Despite the negative I scores for purely artificial datasets, which suggest that they make the original text worse, systems trained on EW-1-POS-U and SW-0-POS-O show an improvement in terms of GLEU, although again this is not significant.

Given the considerable similarity between CLC-train and artificial datasets replicating the original error type distribution (see Table 5.4), it seems surprising that their performance is not close to that of CLC-train. The most likely reason for this is the difference in genre, topic and language complexity, although the latter was expected to be matched by the use of Simple English Wikipedia.

Hybrid datasets, on the contrary, show a consistent improvement in P, $F_{0.5}$, GLEU and I over both the baseline and the original text. This seems to be due to significant increases in P (of up to 5 points or 10.40% in the best case), which translates into improvements in the rest of the measures. This increase in P comes at the expense of R, confirming our earlier results on probabilistic AEG presented in Section 4.3. The higher scores for hybrid datasets also confirm that artificial data is complementary to real learner data, although only in terms of correction confidence

Dataset	P	R	F_1	$F_{0.5}$	GLEU	I
Original text	—	0.00	—	—	60.39	0.00
CLC-train	48.67	37.64	42.45	45.98	67.70	−3.33
EW-0-POS-O	24.51	16.84	19.96	22.46	57.09	−9.63
EW-0-POS-U	25.01	10.84	15.12	19.83	59.01	−5.44
EW-0-POS-S	24.97	10.84	15.12	19.81	58.86	−5.47
EW-1-POS-O	22.98	14.25	17.59	20.47	56.98	−8.70
EW-1-POS-U	30.94	8.98	13.92	20.78	60.72	**−2.98***
EW-1-POS-S	25.32	10.55	14.89	19.78	59.14	−5.26
EW-2-POS-O	28.29	6.92	11.12	17.49	60.16	**−2.68***
EW-2-POS-U	25.19	7.16	11.15	16.75	59.73	−3.44
EW-2-POS-S	22.81	6.55	10.18	15.24	59.34	−3.80
SW-0-POS-O	25.57	14.98	18.89	22.40	65.78	−7.99
SW-0-POS-U	26.50	7.93	12.21	18.05	59.93	−3.56
SW-0-POS-S	28.72	8.58	13.21	19.54	59.99	−3.42
SW-1-POS-O	25.91	11.23	15.67	20.54	58.91	−5.61
SW-1-POS-U	23.45	8.12	12.06	17.02	59.08	−4.57
SW-1-POS-S	26.38	8.71	13.10	18.77	59.66	−4.07
SW-2-POS-O	21.28	6.42	9.86	14.55	58.84	−4.15
SW-2-POS-U	17.24	6.13	9.04	12.65	58.22	−5.71
SW-2-POS-S	18.17	6.31	9.37	13.21	58.54	−5.29
CLC-train+EW-0-POS-O	**52.26***	34.16	41.31	**47.25***	**68.15***	**−1.70***
CLC-train+EW-0-POS-U	**52.85***	32.93	40.58	**47.15***	**68.23***	**−1.29***
CLC-train+EW-0-POS-S	**53.10***	33.02	40.72	**47.34***	**68.32***	**−1.29***
CLC-train+EW-1-POS-O	**52.32***	33.85	41.11	**47.17***	**68.23***	**−1.56***
CLC-train+EW-1-POS-U	**53.73***	32.95	40.85	**47.71***	**68.41***	**−1.12***
CLC-train+EW-1-POS-S	**53.59***	33.00	40.85	**47.64***	**68.36***	**−1.10***
CLC-train+EW-2-POS-O	**52.07***	33.85	41.03	**47.01***	**68.13***	**−1.75***
CLC-train+EW-2-POS-U	**51.80***	34.40	41.34	**47.04***	**68.28***	**−1.77***
CLC-train+EW-2-POS-S	**51.68***	34.09	41.08	**46.85***	**68.21***	**−1.80***
CLC-train+SW-0-POS-O	**49.77***	36.02	41.79	**46.24***	**67.82***	−2.84
CLC-train+SW-0-POS-U	**50.39***	34.05	40.64	45.98	**67.78***	**−2.33***
CLC-train+SW-0-POS-S	**50.49***	34.11	40.71	**46.07***	**67.82***	**−2.29***
CLC-train+SW-1-POS-O	**49.82***	35.65	41.56	**46.15***	**67.88***	−2.65
CLC-train+SW-1-POS-U	**50.47***	35.45	41.65	**46.53***	**68.07***	−2.45
CLC-train+SW-1-POS-S	**50.44***	35.19	41.46	**46.42***	**68.05***	**−2.35***
CLC-train+SW-2-POS-O	**49.45***	35.56	41.37	45.87	**67.83***	−2.65
CLC-train+SW-2-POS-U	**49.74***	35.78	41.62	**46.14***	**68.00***	−2.60
CLC-train+SW-2-POS-S	**49.61***	35.76	41.56	**46.04***	**67.98***	−2.58

Table 5.5: Results of PoS-based probabilistic datasets on FCE-test. Improvements over the baseline (CLC-train) are marked in bold; statistically significant differences are marked with an asterisk.

rather than coverage. Given the general preference for high-precision GEC systems, this is a positive outcome; however, it is still surprising that the use of PoS-based error patterns does not help recall. Values of P over 50 and GLEU scores greater than the original text seem to indicate that many hybrid sets do improve it, although this is clearly not captured by the more conservative I-measure.

The effect of each tested variable on system performance is discussed in detail in the following sections.

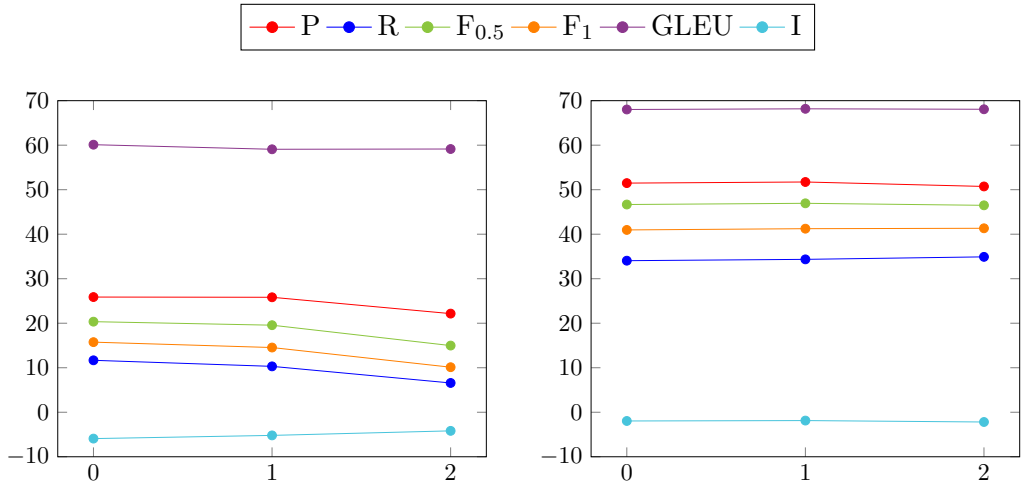
5.2.1 Base texts

In Section 4.3 we commented on the importance of using base texts that are as similar as possible to the reference corpus, so we decided to test this hypothesis by using base texts from two different sources: English Wikipedia and Simple English Wikipedia. While the English Wikipedia is mostly written by and intended for native speakers of the language, its Simple English version is aimed at 'people with different needs, such as students, children, adults with learning difficulties, and people who are trying to learn English' (Wikipedia, 2015), so it appears more suitable for our purpose. Articles written in 'Simple English' employ a simplified form of the English language, typically using the 1,000 most common words and simpler grammatical structures, such as shorter sentences.

A preliminary analysis based on JSD in Table 5.4 does not reveal any noticeable difference between datasets built from these two Wikipedia versions, although such values reflect divergence in error distributions, not language complexity. The performance of purely artificial datasets does not reveal a clear difference either. Simple English datasets perform better 5 out of 9 times in terms of P, GLEU and I; however, the differences are very small and average performance is generally lower than for standard English datasets. The latter outperform Simple English sets consistently on R and F_1, although the difference is often very small too. When used in conjunction with CLC-train, all datasets based on English Wikipedia are superior in terms of P, $F_{0.5}$, GLEU and I. All these systems consistently outperform the baseline on the said metrics, with statistically significant improvements in P and I for all cases.

In conclusion, we found that Simple English datasets seem slightly advantageous when used in isolation while English Wikipedia datasets are clearly superior when combined with genuine learner data. We believe this is because errors injected on similar data cannot encode much more information than what is already present in our reference corpus, while less similar data can provide new contexts which can complement the reference corpus (as confirmed by differences in R).

Given the superior performance of hybrid systems using English Wikipedia datasets, we conclude that this is the most convenient source of base texts.



(a) Purely artificial datasets

(b) Hybrid datasets

Figure 5.3: Effect of context window size on the average performance of evaluation measures.

5.2.2 Context window

We defined a maximum context window of up to 2 tokens to the left and right of an error during the pattern extraction process, which must be matched during error generation. Start and end-of-sentence markers are also included but not counted towards the context window size.

The amount of context used by the error patterns has a direct impact on the quality of the generated errors, as it controls how realistic they look in a given context. Thus, the more context is used, the more natural the errors will look, as demonstrated by the examples in Appendix G.

While the use of larger contexts will produce higher-quality data, it can significantly reduce the number and variety of the generated errors, as base texts are unlikely to exhibit all the required contexts. For example, a pattern with a context window of 2 tokens surrounding an error that involves 3 words will require matching $2+3+2=7$ tokens in a candidate sentence in order to be applied, e.g. */NN1 */TO *catch*/VV0 *up*/RP *on*/II */AT */NN1 → */NN1 */TO *cover*/VV0 */AT */NN1. On the contrary, using no context will generate more errors (and usually more than one in a single sentence) but they are unlikely to resemble real learner errors (e.g. *The port handles all of the international trade for the country.* → **In port handles of the world wide trades because a country.*).

The effect of context length varies widely across datasets and metrics, so a trend might not be obvious. To facilitate analysis, we compute the average performance of each metric for each maximum context window size. This is done separately for purely artificial and hybrid datasets, whose results are plotted in Figure 5.3.

On purely artificial datasets, performance tends to decrease for all metrics as more context is added, although differences are small, especially between 0 and 1.

Dataset	Sentence error rate	Average errors per sentence	JSD
EW-0-LEX-O	61.41%	2.37	0.0069
EW-1-LEX-O	61.41%	1.08	0.0303
EW-2-LEX-O	61.41%	1.00	0.0228
SW-0-LEX-O	61.41%	2.42	0.0064
SW-1-LEX-O	61.41%	1.11	0.0171
SW-2-LEX-O	61.41%	1.00	0.0339

Table 5.6: Error statistics for lexical probabilistic datasets based on CLC-train.

Only I-measure scores show increasing performance. On hybrid datasets, we observe that performance generally increases from 0 to 1 and then drops at 2. Only R and F_1 show a sustained increase, although in all cases differences are very subtle to be perceived in the plot.

The difference in behaviour between these two groups is likely to be caused by the presence of compatible contexts in the data, which are directly related to the nature of texts. Clearly, Wikipedia articles exhibit different long structures than our reference learner data, which is why longer contexts perform poorly with artificial data alone. In particular, the effect of genre becomes more evident, since most long patterns extracted from CLC-train refer to typical phrases used in correspondence, such as salutations and the use of the first person, all highly unlikely to appear in Wikipedia articles (*I would be grateful if you could...*, *write to me soon*, etc.). The addition of learner data clearly makes up for this deficit, as evidenced by hybrid datasets.

Overall, we find that using a maximum context window of 1 token is the most beneficial, as it achieves a good balance between the number of generated errors and how plausible they look in context. The best performing systems, based on CLC-train plus EW-1-POS-U or EW-1-POS-S, seem to support our conclusion.

5.2.3 Lexical vs. PoS patterns

In order to investigate the effect of using lexical error patterns in lieu of PoS-based ones, we generated a small number of artificial datasets using lexical patterns and the original error distribution. As with PoS-based datasets, we computed some basic error statistics as well as JSD between them and CLC-train, which are reported in Table 5.6. A comparison with their PoS-based counterparts in Table 5.4 reveals slightly lower error rates for lexical datasets (except for EW-0-LEX-O) as well as greater JSD (except for EW-2-LEX-O).

As in previous experiments, we used these datasets in isolation and in conjunction with CLC-train to build SMT systems for error correction, which were later evaluated on FCE-test. Results of evaluation are reported in Table 5.7.

Dataset	P	R	F_1	$F_{0.5}$	GLEU	I
Original text	—	0.00	—	—	60.39	0.00
CLC-train	48.67	37.64	42.45	45.98	67.70	−3.33
EW-0-LEX-O	24.90	19.68	21.98	23.65	56.43	−11.10
EW-1-LEX-O	30.16	14.08	19.20	24.55	60.23	−4.88
EW-2-LEX-O	30.51	6.00	10.03	16.79	60.40	**−1.99***
SW-0-LEX-O	24.90	15.77	19.31	22.32	57.59	−8.82
SW-1-LEX-O	27.19	10.60	15.25	20.71	59.87	−4.73
SW-2-LEX-O	23.00	4.66	7.75	12.87	59.74	**−2.79***
CLC-train+EW-0-LEX-O	**51.79***	34.62	41.50	**47.12***	**68.09***	**−1.96***
CLC-train+EW-1-LEX-O	**53.48***	33.50	41.20	**47.78***	**68.45***	**−1.16***
CLC-train+EW-2-LEX-O	**50.89***	35.03	41.50	**46.66***	**68.13***	**−2.03***
CLC-train+SW-0-LEX-O	**49.05***	36.22	41.67	45.80	**67.72***	**−3.04***
CLC-train+SW-1-LEX-O	**49.71***	35.87	41.67	**46.15***	**67.92***	**−2.65***
CLC-train+SW-2-LEX-O	**49.21***	35.95	41.55	45.83	**67.79***	**−2.84***

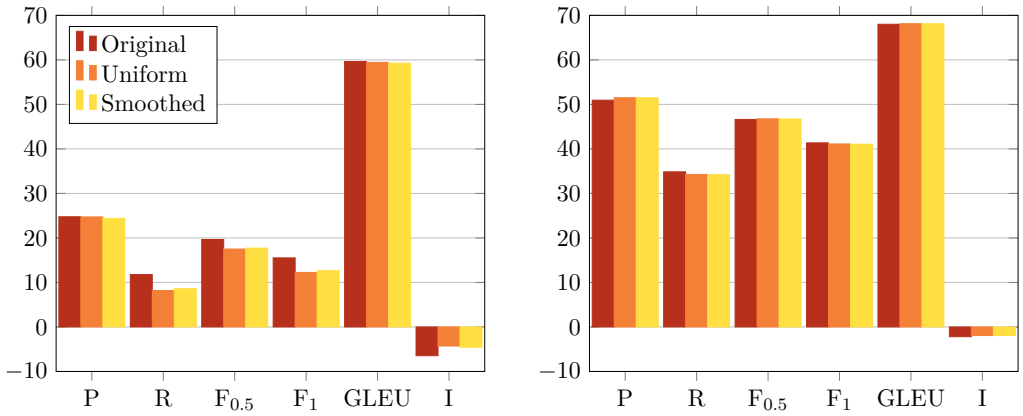
Table 5.7: Results of lexical probabilistic datasets on FCE-test. Improvements over the baseline (CLC-train) are marked in bold; statistically significant differences are marked with an asterisk.

Lexical datasets in isolation show slightly better performance than PoS-based datasets on most metrics, except for R. However, like their counterparts, they also fail to outperform the baseline or improve the original text. By contrast, hybrid datasets show less conclusive results, although CLC-train+EW-1-LEX-O stands out as a winner and even beats the best-performing CLC-train+EW-1-POS-U system on most metrics by a very small margin. Lexical patterns seem more effective than PoS patterns when using a context window of up to 1 token, while for 2 tokens the latter type fares better. This is not surprising, since it is easier to match a long sequence of PoS tags than a long sequence of word forms.

In any case, the slight superiority of lexical patterns would indicate that the errors in the test set (FCE-test) occur in practically the same contexts as the ones in CLC-train, so trying to create errors in new contexts using PoS patterns does not bring much benefit. In order to determine if lexical patterns are consistently superior, we must run additional experiments on different test sets, which we present in Section 5.3.

5.2.4 Generation mode

As with context window size, the impact of the generation mode (following the original, uniform or smoothed distribution) also depends on the evaluation metric and type of dataset used (purely artificial or hybrid). We thus follow a similar approach to the one in Section 5.2.2, whereby we analyse the effect of the variable at hand on the average value of each metric. To facilitate comparison, we plot these values in Figure 5.4.



(a) Purely artificial datasets

(b) Hybrid datasets

Figure 5.4: Effect of generation mode on the average performance of evaluation measures.

We find that for purely artificial datasets, preserving the original distribution observed in our reference corpus (CLC-train) yields the best results for most metrics, with the exception of the I-measure. Smoothed and uniform distributions come second and third respectively, except for P and GLEU where this is the opposite. In any case, the difference between these two generation modes is very small on all evaluation metrics. Results on hybrid datasets show a less clear distinction between modes, although smoothed and uniform distributions are slightly superior on all metrics except R and F_1 (something that is counter-intuitive in principle). In most cases, however, differences between these two modes are almost imperceptible.

While the original distribution gives the best results on purely artificial datasets, it fails to outperform the rest of modes on hybrid sets. The most likely reason for this lies in the fact that purely artificial datasets act as a full replacement for CLC-train so they should mimic it as faithfully as possible to perform well on FCE-test. On the contrary, hybrid sets already contain this information, so there is not much to gain by replicating the same errors in similar contexts. Nevertheless, it is particularly interesting that a uniform distribution, which is the least informed method, can give top results, as demonstrated by the system trained on CLC-train+EW-1-POS-U in Table 5.5. A smoothed distribution can also produce competitive results when used in hybrid datasets; however, it did not perform as well as expected, especially in comparison with a uniform distribution. This might be a sign that the smoothing function was too aggressive, to the point of diluting the differences between types and becoming too similar to the uniform distribution.

In view of these results and given the superiority of hybrid datasets, a uniform distribution seems the most convenient generation mode, since it can produce top results with minimal effort.

Dataset	P	R	F_1	$F_{0.5}$	GLEU	I
Original text	—	0.00	—	—	60.39	0.00
100% CLC-train	48.67	37.64	42.45	45.98	67.70	−3.33
50% CLC-train + 50% EW-1-POS-U	**53.41***	30.85	39.11	**46.60***	**67.86***	**−1.18***
50% CLC-train + 50% EW-1-POS-S	**52.77***	31.09	39.13	**46.31***	**67.77***	**−1.27***

Table 5.8: Comparison of hybrid datasets containing 50% sentences from CLC-train and 50% artificial data on FCE-test. Improvements over the baseline (CLC-train) are marked in bold; statistically significant differences are marked with an asterisk. A combination of real and artificial data yields better results on P-based measures than real data alone.

5.2.5 Dataset size

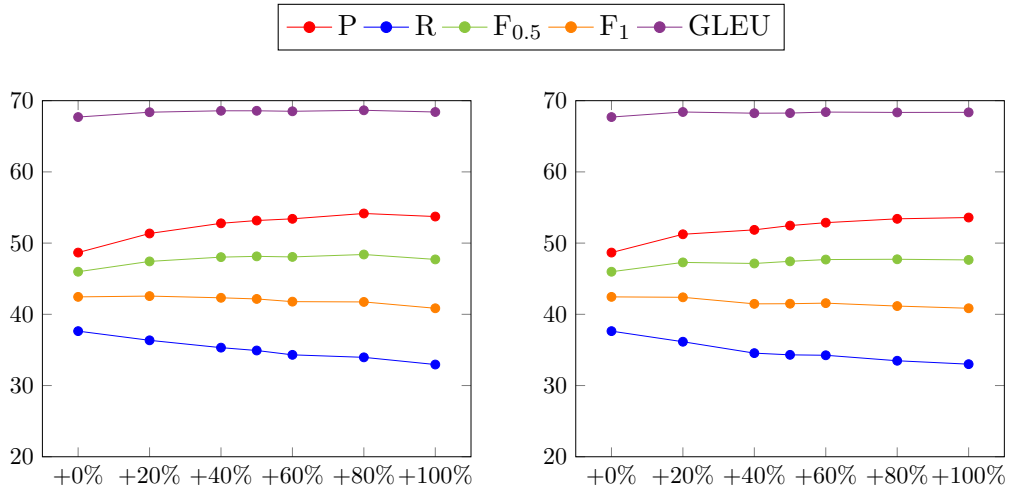
As we have observed, hybrid datasets containing artificial and real learner errors produce a substantial improvement in performance compared to those containing only artificial errors. However, given that the former are twice the size of the latter, a natural question to ask is whether this difference is due to the size of the datasets itself or is indeed the result of their composition (as we have postulated).

The first piece of evidence in support of our thesis is given by a direct comparison between the baseline and systems trained on hybrid datasets. Despite being half the size of the latter, CLC-train produces competitive results that can consistently beat hybrid datasets on certain measures (e.g. R) or show no statistical difference from them.

To further test these claims, we carried out a small experiment where we created two new datasets using the same settings as our best two PoS-based hybrid datasets (based on EW-1-POS-U and EW-1-POS-S). These new sets are the same size as CLC-train (1,965,727 sentences) and include 50% random sentences from that corpus plus 50% random sentences from EW-1-POS-U and EW-1-POS-S respectively. The performance of systems trained on these datasets is reported in Table 5.8. These results show that hybrid sets having the same size as CLC-train can increase the performance on most metrics, with the exception of R and F_1. We thus verify that the composition of the datasets is crucial and conclude that the improvements observed for hybrid datasets in previous experiments are indeed produced by the addition of artificial errors and not a mere by-product of increased size.

We also conducted a study to discover how performance varies as we scale up the training data. Our first experiment investigates variations in performance when artificial data is incrementally added to the baseline system. We do this for the two best-performing types of artificial data (EW-1-POS-U and EW-1-POS-S), which we add to CLC-train in gradual 20% increments. We can see the behaviour of evaluation measures in Figure 5.5.

I-measure scores were found to increase continually in all cases but were excluded from the plot to facilitate visualisation. In general, R and F_1 decrease gradually as more artificial data is added, while P and $F_{0.5}$ increase. For EW-1-POS-U, however,



(a) 100% CLC-train + x% EW-1-POS-U

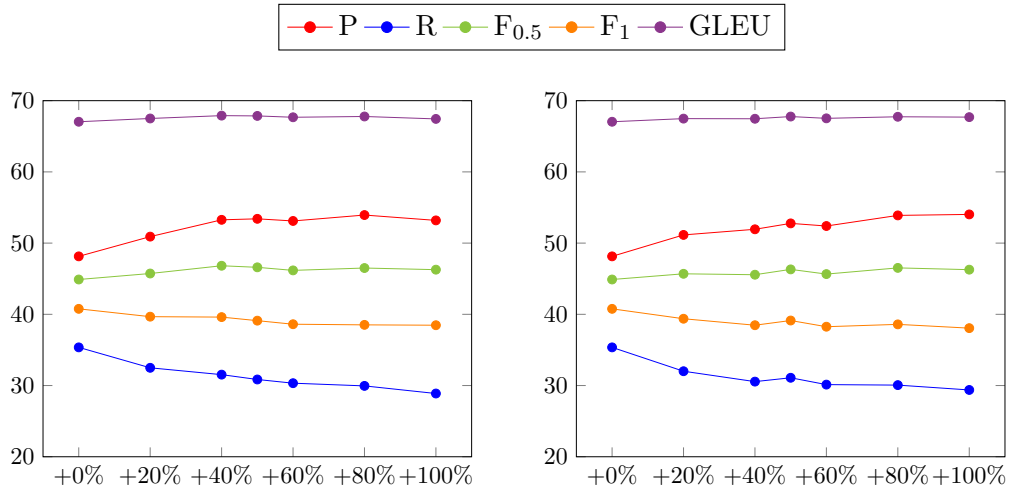
(b) 100% CLC-train + x% EW-1-POS-S

Figure 5.5: System performance when training on 100% CLC-train plus incremental additions of artificial data.

all measures reach their highest at 80% and fall slightly at 100%. The less predictable measure is GLEU, which fluctuates constantly without a clear trend. Overall, these results suggest that keeping a similar proportion of real and artificial errors yields the best results, although the small final decrease for EW-1-POS-U might indicate a slight preference for real errors.

In order to estimate what might happen if we introduced an even greater proportion of artificial errors, we carried out a second experiment. This time, we used a baseline system containing only 50% of CLC-train so as to test performance with up to twice as much artificial data as real data. Results of these experiments are shown in Figure 5.6. Both graphs show similar trends to the ones in Figure 5.5. Since we now use only 50% of CLC-data, the middle point in each graph ($x = 50\%$) represents an equal proportion of real and artificial data; the area to the left shows performance with a lower proportion of artificial data (and is thus comparable to the previous experiment) while the area to the right shows performance with a greater proportion of artificial data. This second area reveals that all measures drop at 60% (i.e. when using 20% more artificial than real data) and then follow the same patterns of behaviour as in our previous experiment: P and $F_{0.5}$ increase, R and F_1 decrease and GLEU fluctuates slightly. In fact, we now observe that most measures suffer from fluctuations, although they follow their previously observed trends.

In conclusion, we confirm that the incremental addition of artificial data can improve P at the expense of R, although P seems to grow at a slightly higher rate if we apply linear regression. This means that P and its related metrics (such as $F_{0.5}$ or I) are likely to increase if we scale up our artificial data beyond the 1:2 ratio.



(a) 50% CLC-train + x% EW-1-POS-U

(b) 50% CLC-train + x% EW-1-POS-S

Figure 5.6: System performance when training on 50% CLC-train plus incremental additions of artificial data.

5.2.6 Upper bounds

As we described in Section 2.1.2.4, an SMT decoder searches the space of possible translations in order to find the best output. As a result of this process, the system creates a list of hypotheses (the n-best list), which ranks them from best to worst according to some specified features (e.g. LM scores), and finally returns the top hypothesis as the output translation (or correction, in our case). In many cases, however, the best corrections (from a human perspective) are not necessarily at the top but lower down the n-best list, and are therefore lost during decoding. This means that a system's correction capability will be underestimated if hypotheses that match the gold standard are not at the top of the n-best list.

To recover these hypotheses and estimate the best possible performance for a system, we can limit the decoder's output to the set of gold-standard corrections. This process, known as *constrained* or *forced decoding*, attempts to find the most likely phrase alignment between the source and target sentences, producing no output if the target cannot be reached. In other words, we can simulate what would happen if we always chose the expected correction from the n-best list (whenever it is available).

The proportion of sentences that are successfully corrected by this 'perfect' system can give us an upper bound on performance and an insight into the usefulness of its phrase translation table. More specifically, if a system trained on dataset A achieves better performance via forced decoding than a system trained on dataset B, we can assume that dataset A is potentially more useful than B, as it allows the system to learn a wider range of corrections.

Computing a true upper bound for each evaluation metric is not as straightforward, however, since it would require choosing the hypotheses that maximise each metric for every sentence in the test set (rather than the 'all or nothing' approach of forced

Dataset	Errors in sentence					Errors in sentence				
	0	**1**	**2**	**3+**	**All**	**0**	**1**	**2**	**3+**	**All**
CLC-train	99.89	83.62	72.39	46.67	78.52	99.89	83.62	72.39	46.67	78.52
	Purely artificial					**Hybrid**				
EW-0-POS-O	**100.00***	57.90	40.22	15.87	59.20	**100.00***	83.48	**73.04***	44.44	78.11
EW-0-POS-U	**100.00***	42.82	25.87	6.03	50.54	**100.00***	83.19	**73.26***	45.08	78.22
EW-0-POS-S	99.67	43.68	26.52	6.35	50.84	**100.00***	83.19	**73.48***	45.08	78.26
EW-1-POS-O	**100.00***	51.01	33.91	9.05	54.74	**100.00***	82.76	71.52	44.76	77.74
EW-1-POS-U	99.78	43.82	26.74	5.56	50.76	**100.00***	83.33	**72.83***	45.24	78.22
EW-1-POS-S	99.78	43.82	26.30	7.30	51.10	**100.00***	82.90	**73.04***	44.92	78.08
EW-2-POS-O	**100.00***	41.38	20.87	3.49	48.72	**100.00***	82.90	72.39	44.29	77.81
EW-2-POS-U	99.89	38.79	20.00	3.33	47.83	**100.00***	83.19	**72.61***	44.44	77.96
EW-2-POS-S	99.89	38.79	19.78	3.17	47.75	**100.00***	83.19	72.17	44.60	77.93
SW-0-POS-O	99.78	59.77	40.87	17.62	60.13	99.78	83.33	71.96	45.24	78.00
SW-0-POS-U	99.89	43.39	26.96	6.67	50.98	**100.00***	83.33	**73.48***	45.08	78.30
SW-0-POS-S	**100.00***	44.68	27.83	6.98	51.58	**100.00***	83.33	**73.48***	44.76	78.22
SW-1-POS-O	**100.00***	51.15	31.30	9.68	54.48	**100.00***	82.90	71.30	45.24	77.85
SW-1-POS-U	99.56	46.26	27.83	6.98	51.84	**100.00***	83.48	**72.61***	46.19	78.45
SW-1-POS-S	99.56	45.69	28.70	6.67	51.77	**100.00***	83.48	71.96	45.87	78.26
SW-2-POS-O	**100.00***	37.64	19.78	3.02	47.45	**100.00***	83.19	72.17	45.24	78.08
SW-2-POS-U	99.67	35.78	19.13	2.70	46.67	**100.00***	83.33	**73.26***	45.40	78.34
SW-2-POS-S	99.78	36.93	18.70	2.54	46.90	**100.00***	83.19	**73.04***	45.08	78.19
EW-0-LEX-O	**100.00***	59.05	44.35	18.41	60.80	**100.00***	83.62	**72.61***	44.60	78.11
EW-1-LEX-O	**100.00***	46.12	29.57	6.98	52.25	**100.00***	82.90	**72.83***	45.40	78.15
EW-2-LEX-O	99.78	28.02	8.26	0.00	42.21	**100.00***	83.33	72.39	45.08	78.11
SW-0-LEX-O	99.78	61.64	44.78	19.68	61.76	**100.00***	83.33	**73.26***	44.92	78.22
SW-1-LEX-O	99.78	47.70	27.61	5.87	51.99	**100.00***	82.61	72.39	45.24	77.96
SW-2-LEX-O	99.56	18.68	5.00	0.00	39.17	99.89	83.62	71.96	46.03	78.30

Table 5.9: Proportion of successfully corrected sentences using forced decoding on FCE-test. Improvements over the baseline (CLC-train) are marked in bold.

decoding). While this can be informative for a small number of systems, it would be impractical in our case, which is why we opted for comparisons in terms of proportions.

We performed forced decoding on FCE-test for all our systems and reported results in Table 5.9, according to the number of errors per sentence. A baseline system trained on CLC-train is potentially able to identify almost all the sentences that do not need correction (99.89%) and correct erroneous sentences in diminishing proportions as the number of errors in a sentence increases. The overall potential correction rate, i.e. the proportion of incorrect sentences for which a gold standard correction is in the n-best list, is 78.52%.

Systems trained on artificial data exhibit generally low performance, which decreases dramatically as the number of errors increases. Nevertheless, some datasets (especially those based on the original error distribution) can achieve a perfect detection rate for correct sentences. As in the rest of experiments, hybrid systems show better performance and can not only identify correct sentences in almost all cases but also improve correction for sentences containing only 2 errors. Given that the average number of errors per sentence in most of our datasets is closer to 1, we would have expected this category to show improvements instead.

It is particularly surprising that systems trained on hybrid datasets are unable to outperform the baseline, since all the phrase alignments extracted from CLC-train should also be present. However, as noted by Foster and Kuhn (2012), forced decoding is also subject to the usual decoding heuristics (word penalty, stack size, etc.) because it is impossible to search the entire space exhaustively. This means that even if an alignment is possible with a given phrase table, the decoder might not find it, resulting in decoding failures. In fact, the authors show that it is not always possible to 'forced decode' every sentence in a training corpus using a phrase table extracted from it, and report failure rates between 20% and 50% on different corpora.

Based on these findings, we argue that the addition of more training samples can only be beneficial to SMT-based GEC if the incorrect phrases and their corrections occur frequently; otherwise, they will just introduce noise and hinder the decoding process. Given the highly skewed distribution in the CLC-train corpus, this might be one of the reasons why uniform and smoothed datasets achieve better performance.

Another likely source of confusion is the creation of artificial errors using patterns that can actually produce correct language, especially if they are too general or use a small context window. For example, an error generation pattern such as *could* → *can*, which is completely devoid of context, could generate a perfectly valid sentence that will be incorrectly flagged as erroneous. Most of these cases can probably be avoided by filtering out such patterns prior to error generation or using a LM to estimate the probability of artificial sentences being correct before adding them to the corpus. Alternatively, to recover good correction hypotheses that are buried in the n-best list during decoding, we could perform hypothesis re-ranking on the first n entries of the list using an external model (e.g. a large LM or a classifier trained on a variety of features), so that the best candidate has better chances of emerging to the surface (Felice et al., 2014; Zhao et al., 2015). Our experiments using a re-ranking model based on novel statistical features (such as SMT decoder scores and detailed LM statistics) showed that we can increase I-measure performance on FCE-test by up to 9.78% while boosting $F_{0.5}$ (Yuan et al., 2016). The calculated upper bound for the re-ranking task achieves over 40% I-measure performance, demonstrating that there is considerable room for improvement.

5.3 Comparison with systems in the CoNLL-2014 shared task

As explained in Section 5.1.1, the FCE-test set used in our experiments is a publicly available portion of the CLC which is considered to be representative of upper-intermediate learner errors. Although FCE-test is obviously not part of CLC-train, it is very likely that there is a significant overlap of errors between them, as both datasets are drawn from the same corpus. Since our artificial datasets were based on

errors in CLC-train, we expected them to perform well on the test set; however, this does not reveal if artificial errors can help correct real errors from a different corpus of similar exam scripts. To test this, we evaluated our systems on the CoNLL-2014 shared task test set, which also contains learner sentences with a variety of errors (see Section 2.3.2). As in Section 4.4.1, we evaluated our systems using the official M^2 Scorer and analysed how they would rank among the 13 participating teams. System performance is measured in terms of $F_{0.5}$, as it was the official evaluation metric adopted in the shared task.

Results are reported in Table 5.10. There are two test sets: an 'original' one containing corrections from the two official annotators and an 'alternative' one including additional annotations provided by three participating teams. A baseline system trained on CLC-train shows much lower performance on the CoNLL-2014 test set than on FCE-test, with a drop in $F_{0.5}$ of 29.25% (from 45.98 in Table 5.5 to 32.53 in Table 5.10) and 24.64% (from 45.98 in Table 5.5 to 34.65 in Table 5.10) on the original and alternative test sets respectively. These results indicate that errors on both test sets are considerably different, which is not surprising, since essays in the CoNLL-2014 test set are indeed from a more specific population (advanced learners with mostly Asian L1s).

Systems trained on purely artificial datasets also show poor performance but, unlike the baseline, their $F_{0.5}$ scores only drop by an average of 3.18% on the original test set while they increase by an average of 8.14% on the alternative test set with respect to performance on FCE-test (i.e. average per cent difference between scores in Table 5.5 and Table 5.10). This suggests that artificial datasets are more robust and stable than real learner data across different test sets (as they include errors in a wider variety of contexts than CLC-train) and also that Wikipedia articles are more similar to the CoNLL-2014 test set. Our conclusions about the robustness of artificial data seem to confirm the findings by Cahill et al. (2013b).

When hybrid datasets are used, performance rises considerably on both test sets. Improvements over the baseline range between 4.5% and 17%, and are statistically significant in most cases. On the original test set, most systems would rank within the top three while three of them (EW-0-POS-S, EW-1-POS-S and EW-0-POS-U) would rank first. This provides evidence in favour of one of our key research questions: can we improve performance by augmenting the training data instead of doing system engineering? On the alternative test set, ranks go down in general, but this is because $F_{0.5}$ scores for the participating teams increased in higher proportions.

Results on these test sets also allow us to revisit our findings on the best settings for probabilistic AEG. While we can confirm our conclusions about English Wikipedia, context length and pattern type, we now observe that smoothed datasets can slightly outperform uniform datasets in hybrid scenarios. However, they are still strong contenders: while CLC-train+EW-1-POS-S wins on the original test set, CLC-train+EW-1-POS-U is better on the alternative set.

Dataset	Original test set				Alternative test set			
	$F_{0.5}$	Rank	$F_{0.5}$	Rank	$F_{0.5}$	Rank	$F_{0.5}$	Rank
CLC-train	32.53	5	32.53	5	34.65	6	34.65	6
	Purely art.		**Hybrid**		**Purely art.**		**Hybrid**	
EW-0-POS-O	19.53	10	**36.47***	**3**	21.49	10	**38.70***	**3**
EW-0-POS-U	19.17	10	**36.25***	**3**	20.96	10	**38.64***	**3**
EW-0-POS-S	20.66	10	**37.69***	**1**	22.46	10	**40.10***	**3**
EW-1-POS-O	20.00	10	**35.82***	**3**	22.12	10	**38.48***	**4**
EW-1-POS-U	18.48	10	**37.62***	**1**	21.18	10	**40.45***	**3**
EW-1-POS-S	19.93	10	**38.05***	**1**	21.90	10	**40.38***	**3**
EW-2-POS-O	16.90	10	**37.15***	**2**	19.95	10	**39.82***	**3**
EW-2-POS-U	17.08	10	**36.39***	**3**	18.79	10	**38.50***	**4**
EW-2-POS-S	15.90	10	**35.96***	**3**	17.97	10	**38.18***	**4**
SW-0-POS-O	20.63	10	**34.58***	**4**	20.63	10	**36.99***	**4**
SW-0-POS-U	17.56	10	**35.97***	**3**	19.65	10	**38.27***	**4**
SW-0-POS-S	16.12	10	**36.01***	**3**	16.12	11	**38.31***	**4**
SW-1-POS-O	20.27	10	**35.96***	**3**	22.73	10	**38.43***	**4**
SW-1-POS-U	17.46	10	**35.67***	**3**	19.71	10	**37.88***	**4**
SW-1-POS-S	16.48	10	**36.33***	**3**	19.00	10	**38.45***	**4**
SW-2-POS-O	16.85	10	**34.02***	**4**	18.64	10	**36.14***	**5**
SW-2-POS-U	14.17	11	**35.17***	**3**	16.14	11	**37.34***	**4**
SW-2-POS-S	13.81	11	**34.97***	**4**	15.58	11	**37.17***	**4**
EW-0-LEX-O	23.37	9	**36.73***	**3**	25.38	9	**39.20***	**3**
EW-1-LEX-O	20.33	10	**36.90***	**2**	22.87	10	**39.68***	**3**
EW-2-LEX-O	14.99	11	**34.56***	**4**	17.08	11	**36.73***	**5**
SW-0-LEX-O	22.17	10	**34.89***	**4**	24.36	10	**37.26***	**4**
SW-1-LEX-O	17.79	10	**35.56***	**3**	21.00	10	**38.11***	**4**
SW-2-LEX-O	12.35	10	**34.15***	**4**	15.20	11	**36.16***	**5**

Table 5.10: Rankings of our systems using probabilistic artificial datasets with respect to the 13 participating systems in the CoNLL-2014 shared task. Improvements over the baseline (CLC-train) are marked in bold; statistically significant differences are marked with an asterisk. Three of the systems trained on real and artificial data are able to beat all the shared task systems.

To sum up, results on the CoNLL-2014 test set show that systems trained on our artificial errors can generalise well and achieve state-of-the-art performance on a different corpus of learner writing. As a result, we can also conclude that the performance on FCE-test is not affected by overfitting.

Chapter 6

Conclusions

This work has focused on the creation of artificial errors to support GEC for language learners. These errors are based on patterns observed in a reference corpus of learner writing and injected into error-free text using different methods. These new texts can then be used as parallel data to train SMT systems that translate from 'incorrect' into 'correct' English. As described in Chapter 2, neither SMT nor the creation of artificial errors are new for GEC; however, their use has been limited to the correction of only a few closed-class error types. While SMT has become a popular framework for error correction and grown to embrace more sophisticated techniques and error types, the adoption of artificial errors has been more limited and less guided. Researchers have used different AEG techniques for different purposes, with no systematic studies on their use in SMT-based GEC. The work presented in this thesis has aimed to help fill this gap.

We also observed a similar scenario concerning the evaluation of GEC systems, where a whole host of different measures have been used in the literature without a clear consensus on the most appropriate for the task. We reviewed these measures in Chapter 3 and discussed many issues with existing approaches, in particular, the popular M^2 Scorer adopted in the CoNLL shared tasks. This analysis led to the first of our contributions: the I-measure, a new evaluation scheme that addresses many of the problems with previous metrics (Section 3.2). Some of the advantages of our measure include the use of tokens as a stable unit of evaluation, the ability to mix and match corrections from different annotators which avoids underestimations of performance, and the computation of scores for detection and correction.

The I-measure quantifies the relative improvement of a system's proposed corrections over the original text, giving a clearer interpretation of system performance. Unlike other metrics, it takes TNs explicitly into account and allows for different weights of TNs, TPs, FNs and FPs. This enables us to tune evaluation to specific needs, for example, by being more or less severe towards missed or unnecessary corrections. A comparison of the I-measure with other scorers revealed significant differences when ranking systems in the CoNLL-2014 shared task and showed that

our measure with default parameters is somewhat conservative. This brings to the fore essential questions about evaluation for GEC for which there are no definitive answers. For example, it is not clear if tokens or phrases are more appropriate units of evaluation, as humans are not likely to think in terms of fixed units when judging language. People have different perceptions of correctness so the idea of a perfect measure is elusive. However, we believe new metrics should follow up on recent proposals that take this variability into account, for example, by comparing system performance with a human upper bound or using agreement to weight errors.

Likewise, there is no clear consensus on whether P should be prioritised over R in general (and, in consequence, use $F_{0.5}$ or F_1), although we argue in favour of this position in line with latest research. Another crucial issue is the limitation that gold standard references impose on the measures, whereupon system corrections which deviate from the references will be considered incorrect even if they are valid. In this case, the use of reference-free measures similar to the ones for quality estimation of MT output may provide an alternative.

In Chapter 4 we presented experiments on the reduced set of error types used in the CoNLL-2013 shared task, involving articles and determiners, noun number, prepositions, subject-verb agreement and verb forms. We proposed and compared two types of methods: random and probabilistic. In both cases, errors are injected into error-free text according to patterns observed in a reference corpus. However, while random generation inserts errors arbitrarily, probabilistic methods follow the original distribution observed in the reference corpus. We proposed five probabilistic datasets based on different types of information: error type distributions, morphology, PoS disambiguation, semantic classes and word senses. This is the first time that this type of linguistic information has been used to generate artificial errors.

Our experiments revealed that these two methods perform differently: while random generation increases R and decreases P, probabilistic generation does the opposite. This is not very surprising, since generating errors at random is likely to produce errors in new contexts and achieve more coverage, while following the same distributions as in the reference corpus will make a system more confident in flagging known errors. We also found that the least sophisticated information (error type distributions and PoS disambiguation) were the most useful in general, outperforming more elaborate linguistic information. In addition, an error type analysis revealed that performance tends to decrease with bigger confusion sets.

Results also showed a trend that we verified in later experiments: artificial errors alone are not able to replace real learner data but they can give the best results if used in combination. This was the first confirmation of our hypothesis which stated we could improve performance by tuning the training data instead of system parameters. A comparison with systems in the CoNLL-2013 shared task confirmed these results and demonstrated that a system trained on real and artificial errors can outperform others that use carefully engineered architectures.

Given these encouraging results, we extended AEG to all error types in Chapter 5, as a means to improve general error correction. To the best of our knowledge, this is the first time that artificial errors have been generated for all types, since all previous approaches only worked on a limited set (typically articles and prepositions). These new experiments were based on probabilistic approaches only, as they achieved the best results in terms of P and other measures that we consider more important for the aims of GEC. Additionally, we refined our generation method to produce more accurate errors based on new statistics from the reference corpus (such as the proportion of incorrect sentences and the number of errors per sentence) and investigated a number of variables, including the source of base texts (English Wikipedia vs. Simple English Wikipedia), context window size (0, 1 and 2), type of error patterns (PoS-based vs. lexical) and generation mode (i.e. error distribution: original, uniform and smoothed).

Experiments on FCE-test revealed that the best results are obtained when we generate errors using English Wikipedia and a context window of size 1. However, optimal values for the rest of variables are not clear-cut: lexical patterns and uniform distributions can produce the best results but PoS-based patterns and smoothed distributions are more robust and generalise better to other datasets. We also confirmed that hybrid datasets combining artificial and real errors yield the best performance and found that the addition of more artificial errors is increasingly beneficial. Experiments with up to $2x$ artificial errors showed improvements on all P-based measures and the observed trend suggests that this would hold for higher factors.

A further study where we analysed the upper bounds of a baseline system trained on real learner errors vs. systems incorporating artificial data showed that hybrid systems can improve the detection of correct sentences and the correction of sentences with only two errors. Finally, a comparison of our systems with submissions to the CoNLL-2014 shared task showed that three of our hybrid systems can rank first on the original test set while a baseline system trained on real learner data alone can only achieve a fifth place. This reveals that hybrid systems can generalise well and confirms that the addition of artificial errors can improve performance without having to tune systems.

As a result, we conclude that the generation of artificial errors is a useful technique to overcome the problem of insufficient data for GEC which allows errors to be tailored to specific needs. Our work has shown positive results for all our research goals, demonstrating that AEG (and probabilistic AEG in particular) can be successfully extended to all error types and used to increase performance without the need for system engineering.

There are of course a number of issues that lend themselves to further investigation. Despite our efforts to provide a better evaluation scheme, the I-measure seems very conservative so it can often contradict other measures and thus be seen as unreliable.

As noted previously, this is likely to be the result of using predefined unoptimised weights for each count, so we believe that optimising these weights is essential to test the true capabilities of our measure. This effect can be minimised with the use of alternative corrections as shown in Section 3.2.1, so we encourage the use of multiple corrections whenever possible. At the same time, we firmly believe that other aspects of correction should be considered in new evaluation measures, such as the salience and importance of errors, dependencies between corrections and the effect of system corrections on the original text. Another aspect that deserves more attention is error detection, which we have not reported in our experiments to avoid unfair comparisons with the rest of the metrics. Error detection can be more useful than correction in many cases, so performance on this aspect is very informative.

We also believe that evaluating performance by error type is very useful and would facilitate comparison between systems. While this is relatively easy to do for a limited number of error types (as in Chapter 4), it can be cumbersome for general error correction, since datasets often use different error typologies and can define a large number of types. Ideally, new evaluation methods should classify corrections according to a dataset-independent typology and report individual results by type.

As for the AEG methods themselves, we should investigate new parameters that could allow us to increase R without harming P. For example, we could integrate base texts from different sources to cover a wider range of vocabulary and structures, extract and apply more general PoS-based error generation patterns and use a LM to determine if a sentence containing artificial errors is indeed likely to be incorrect before adding it to the corpus. Another alternative is to relax some of the constraints in the probabilistic generation process that might have an effect on the number of generated errors, such as the restriction on the number of errors per sentence or the proportion of incorrect sentences in the corpus. To minimise differences between the base texts and the test set, we could attempt to re-use the corrected learner sentences in the reference corpus as additional base texts (as in Foster and Andersen (2009)), so that we can generate new errors in more similar contexts. We should also study if the frequency of error patterns in the corpus has an impact on performance, since we only experimented with a threshold of 10 minimum occurrences. The use of different smoothing functions could also be explored, especially since they can be implemented with ease.

We have also shown that system performance increases as more artificial data is used for training. However, we only verified this for hybrid datasets using up to $2x$ artificial data due to practical limitations, from which we estimated that larger datasets are likely to follow this trend and increase performance. Future work should investigate how this verifies in reality and evaluate datasets which are order of magnitudes larger, e.g. by 10 or 100 times.

Finally, we believe an in-depth study of particular SMT parameters is essential to make the most of the training data. However, because our work only explored

the effects of using artificial errors as training data and not the adaptation of SMT to GEC per se, we have not focused on any of these parameters. For example, we could conduct studies using different alignment models, larger LMs, additional decoding scores, parameter tuning, etc. In fact, recent work has shown that adding Levenshtein distance as an extra score during the decoding phrase can improve performance (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014) and that parameter tuning based on evaluation measures intended for GEC can also lead to improvements (Junczys-Dowmunt and Grundkiewicz, 2014). SMT has grown to become a state-of-the-art approach to GEC but we still have a long way to go before we have exhausted all its potential.

Appendices

Appendix A

CLC error taxonomy

AG	Agreement error
AGA	Anaphor agreement error
AGD	Determiner agreement error
AGN	Noun agreement error
AGQ	Quantifier agreement error
AGV	Verb agreement error
AS	Argument structure error
C	Countability error
CD	Wrong determiner because of noun countability
CE	Complex error
CL	Collocation or tautology error
CN	Countability of noun error
CQ	Wrong quantifier because of noun countability
DA	Derivation of anaphor error
DC	Derivation of link word error
DD	Derivation of determiner error
DI	Incorrect determiner inflection
DJ	Derivation of adjective error
DN	Derivation of noun error
DQ	Derivation of quantifier error
DT	Derivation of preposition error
DV	Derivation of verb error
DY	Derivation of adverb error
FA	Wrong anaphor form
FC	Wrong link word form
FD	Incorrect determiner form
FJ	Wrong adjective form
FN	Wrong noun form
FQ	Wrong quantifier form
FT	Wrong preposition form
FV	Wrong verb form
FY	Wrong adverb form
IA	Incorrect anaphor inflection
ID	Idiom wrong
IJ	Incorrect adjective inflection
IN	Incorrect noun inflection

IQ	Incorrect quantifier inflection
IV	Incorrect verb inflection
IY	Incorrect adverb inflection
L	Inappropriate register
M	Missing error
MA	Missing anaphor
MC	Missing link word
MD	Missing determiner
MJ	Missing adjective
MN	Missing noun
MP	Missing punctuation
MQ	Missing quantifier
MT	Missing preposition
MV	Missing verb
MY	Missing adverb
NE	No error
R	Replace error
RA	Replace anaphor
RC	Replace link word
RD	Replace determiner
RJ	Replace adjective
RN	Replace noun
RP	Replace punctuation
RQ	Replace quantifier
RT	Replace preposition
RV	Replace verb
RY	Replace adverb
S	Spelling error
SA	Spelling American
SX	Spelling confusion
TV	Incorrect tense of verb
U	Unnecessary error
UA	Unnecessary anaphor
UC	Unnecessary link word
UD	Unnecessary determiner
UJ	Unnecessary adjective
UN	Unnecessary noun
UP	Unnecessary punctuation
UQ	Unnecessary quantifier
UT	Unnecessary preposition
UV	Unnecessary verb
UY	Unnecessary adverb
W	Word order error
X	Incorrect negative formation

Appendix B

NUCLE error taxonomy

ArtOrDet	Article or determiner
Cit	Citation
Mec	Spelling, punctuation, capitalization, etc.
Nn	Noun number
Npos	Noun possessive
Others	Other errors
Pform	Pronoun form
Pref	Pronoun reference
Prep	Preposition
Rloc-	Redundancy
Sfrag	Sentence fragment
Smod	Dangling modifiers
Spar	Parallelism
Srun	Run-on sentences, comma splices
Ssub	Subordinate clause
SVA	Subject-verb agreement
Trans	Linking words/phrases
Um	Unclear meaning
V0	Missing verb
Vform	Verb form
Vm	Verb modal
Vt	Verb tense
Wa	Acronyms
Wci	Wrong collocation/idiom
Wform	Word form
WOadv	Incorrect adjective/adverb order
WOinc	Incorrect word order
Wtone	Tone (formal/informal)

Appendix C

Example I-measure annotation for the CLC

Original CLC annotation

```
Since the <NS type="RP"><i>internet</i><c>Internet</c></NS> was
introduced, many of us <NS type="TV"><i><NS type="RV"><i>wouldn't
have imagined</i><c>couldn't have imagined</c></NS></i><c>can't
imagine</c></NS> <NS type="UD"><i>the</i></NS> <NS type="FN"><i>
<NS type="IN"><i>lifes</i><c>lives</c></NS></i><c>life</c></NS>
without <NS type="RA"><i>this</i><c>it</c></NS>.
```

Equivalent I-measure annotation

```
<sentence id="1" numann="1">
  <text>
    Since the internet was introduced , many of us would n't have
    imagined the lifes without this .
  </text>
  <error-list>
    <error id="1" req="yes" type="RP">
      <alt ann="0">
        <c start="2" end="3">Internet</c>
      </alt>
    </error>
    <error id="2" req="yes" type="TV+RV">
      <alt ann="0">
        <c start="9" end="13">ca n't imagine</c>
      </alt>
    </error>
    <error id="3" req="yes" type="UD">
      <alt ann="0">
        <c start="13" end="14"/>
      </alt>
    </error>
```

```
      <error id="4" req="yes" type="FN+IN">
        <alt ann="0">
          <c start="14" end="15">life</c>
        </alt>
      </error>
      <error id="5" req="yes" type="RA">
        <alt ann="0">
          <c start="16" end="17">it</c>
        </alt>
      </error>
    </error-list>
  </sentence>
```

Appendix D

Penn Treebank PoS tags

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Appendix E

CLAWS2 PoS tags

!	punctuation tag - exclamation mark
"	punctuation tag - quotation marks
$	germanic genitive marker - (‘ or ’s)
&FO	formula
&FW	foreign word
(	punctuation tag - left bracket
)	punctuation tag - right bracket
,	punctuation tag - comma
-	punctuation tag - dash
-----	new sentence marker
.	punctuation tag - full-stop
...	punctuation tag - ellipsis
:	punctuation tag - colon
;	punctuation tag - semi-colon
?	punctuation tag - question-mark
APP$	possessive pronoun, pre-nominal (my, your, our etc.)
AT	article (the, no)
AT1	singular article (a, an, every)
BCS	before-conjunction (in order (that), even (if etc.))
BTO	before-infinitive marker (in order, so as (to))
CC	coordinating conjunction (and, or)
CCB	coordinating conjunction (but)
CF	semi-coordinating conjunction (so, then, yet)
CS	subordinating conjunction (if, because, unless)
CSA	‘as’ as a conjunction
CSN	‘than’ as a conjunction
CST	‘that’ as a conjunction
CSW	‘whether’ as a conjunction
DA	after-determiner (capable of pronominal function) (such, former, same)
DA1	singular after-determiner (little, much)
DA2	plural after-determiner (few, several, many)
DA2R	comparative plural after-determiner (fewer)
DAR	comparative after-determiner (more, less)
DAT	superlative after-determiner (most, least)
DB	before-determiner (capable of pronominal function) (all, half)
DB2	plural before-determiner (capable of pronominal function) (eg. both)

DD	determiner (capable of pronominal function) (any, some)
DD1	singular determiner (this, that, another)
DD2	plural determiner (these, those)
DDQ	wh-determiner (which, what)
DDQ$	wh-determiner, genitive (whose)
DDQV	wh-ever determiner (whichever, whatever)
EX	existential 'there'
ICS	preposition-conjunction (after, before, since, until)
IF	'for' as a preposition
II	preposition
IO	'of' as a preposition
IW	'with'; 'without' as preposition
JA	predicative adjective (tantamount, afraid, asleep)
JB	attributive adjective (main, chief, utter)
JBR	attributive comparative adjective (upper, outer)
JBT	attributive superlative adjective (utmost, uttermost)
JJ	general adjective
JJ	general comparative adjective (older, better, bigger)
JJT	general superlative adjective (oldest, best, biggest)
JK	adjective catenative ('able' in 'be able to'; 'willing' in 'be willing to')
LE	leading co-ordinator ('both' in 'both...and...'; 'either' in 'either... or...')
MC	cardinal number neutral for number (two, three...)
MC$	genitive cardinal number, neutral for number (10's)
MC-MC	hyphenated number 40-50, 1770-1827)
MC1	singular cardinal number (one)
MC2	plural cardinal number (tens, twenties)
MD	ordinal number (first, 2nd, next, last)
MF	fraction, neutral for number (quarters, two-thirds)
NC2	plural cited word ('ifs' in 'two ifs and a but')
ND1	singular noun of direction (north, southeast)
NN	common noun, neutral for number (sheep, cod)
NN1	singular common noun (book, girl)
NN1$	genitive singular common noun (domini)
NN2	plural common noun (books, girls)
NNJ	organization noun, neutral for number (department, council, committee)
NNJ1	singular organization noun (Assembly, commonwealth)
NNJ2	plural organization noun (governments, committees)
NNL	locative noun, neutral for number (Is.)
NNL1	singular locative noun (street, Bay)
NNL2	plural locative noun (islands, roads)
NNO	numeral noun, neutral for number (dozen, thousand)
NNO1	singular numeral noun (no known examples)
NNO2	plural numeral noun (hundreds, thousands)
NNS	noun of style, neutral for number (no known examples)
NNS1	singular noun of style (president, rabbi)
NNS2	plural noun of style (presidents, viscounts)
NNSA1	following noun of style or title, abbreviatory (M.A.)
NNSA2	following plural noun of style or title, abbreviatory
NNSB	preceding noun of style or title, abbr. (Rt. Hon.)
NNSB1	preceding sing. noun of style or title, abbr. (Prof.)
NNSB2	preceding plur. noun of style or title, abbr. (Messrs.)

NNT	temporal noun, neutral for number (no known examples)
NNT1	singular temporal noun (day, week, year)
NNT2	plural temporal noun (days, weeks, years)
NNU	unit of measurement, neutral for number (in., cc.)
NNU1	singular unit of measurement (inch, centimetre)
NNU2	plural unit of measurement (inches, centimetres)
NP	proper noun, neutral for number (Indies, Andes)
NP1	singular proper noun (London, Jane, Frederick)
NP2	plural proper noun (Browns, Reagans, Koreas)
NPD1	singular weekday noun (Sunday)
NPD2	plural weekday noun (Sundays)
NPM1	singular month noun (October)
NPM2	plural month noun (Octobers)
PN	indefinite pronoun, neutral for number ("none")
PN1	singular indefinite pronoun (one, everything, nobody)
PNQO	whom
PNQS	who
PNQV$	whosever
PNQVO	whomever, whomsoever
PNQVS	whoever, whosoever
PNX1	reflexive indefinite pronoun (oneself)
PP$	nominal possessive personal pronoun (mine, yours)
PPH1	it
PPHO1	him, her
PPHO2	them
PPHS1	he, she
PPHS2	they
PPIO1	me
PPIO2	us
PPIS1	I
PPIS2	we
PPX1	singular reflexive personal pronoun (yourself, itself)
PPX2	plural reflexive personal pronoun (yourselves, ourselves)
PPY	you
RA	adverb, after nominal head (else, galore)
REX	adverb introducing appositional constructions (namely, viz, eg.)
RG	degree adverb (very, so, too)
RGA	post-nominal/adverbial/adjectival degree adverb (indeed, enough)
RGQ	wh- degree adverb (how)
RGQV	wh-ever degree adverb (however)
RGR	comparative degree adverb (more, less)
RGT	superlative degree adverb (most, least)
RL	locative adverb (alongside, forward)
RP	prep. adverb; particle (in, up, about)
RPK	prep. adv., catenative ('about' in 'be about to')
RR	general adverb
RRQ	wh- general adverb (where, when, why, how)
RRQV	wh-ever general adverb (wherever, whenever)
RRR	comparative general adverb (better, longer)
RRT	superlative general adverb (best, longest)
RT	nominal adverb of time (now, tommorow)

TO	infinitive marker (to)
UH	interjection (oh, yes, um)
VB0	be
VBDR	were
VBDZ	was
VBG	being
VBM	am
VBN	been
VBR	are
VBZ	is
VD0	do
VDD	did
VDG	doing
VDN	done
VDZ	does
VH0	have
VHD	had (past tense)
VHG	having
VHN	had (past participle)
VHZ	has
VM	modal auxiliary (can, will, would etc.)
VMK	modal catenative (ought, used)
VV0	base form of lexical verb (give, work etc.)
VVD	past tense form of lexical verb (gave, worked etc.)
VVG	-ing form of lexical verb (giving, working etc.)
VVN	past participle form of lexical verb (given, worked etc.)
VVZ	-s form of lexical verb (gives, works etc.)
VVGK	-ing form in a catenative verb (‘going’ in ‘be going to’)
VVNK	past part. in a catenative verb (‘bound’ in ‘be bound to’)
XX	not, n’t
ZZ1	singular letter of the alphabet:‘A’, ‘a’, ‘B’, etc.
ZZ2	plural letter of the alphabet: ‘As’, b’s, etc.

Appendix F

CLC-train error type statistics

First 100 most frequent error types and their relative frequency in CLC-train.

RP	7.96%	L	0.53%	TV+S	0.10%
S	7.14%	UY	0.52%	RV+S	0.10%
MP	6.72%	MY	0.46%	UQ	0.09%
RT	6.39%	SA	0.42%	RP+RP	0.08%
MD	5.81%	RC	0.41%	R+S	0.08%
TV	5.34%	U	0.41%	RJ+S	0.08%
RV	4.87%	AGA	0.35%	W+S	0.08%
UP	3.27%	UC	0.35%	TV+AGV	0.08%
RN	3.21%	UN	0.34%	S+RP	0.07%
UD	2.70%	CE	0.32%	FN+S	0.07%
MT	2.66%	IN	0.30%	W+RP	0.07%
FV	2.59%	ID	0.29%	RV+AGV	0.07%
R	2.26%	DV	0.29%	RN+MP	0.07%
UT	1.81%	MQ	0.29%	FV+RV	0.07%
MA	1.80%	RQ	0.28%	RN+RP	0.06%
FN	1.70%	AGD	0.28%	FV+S	0.05%
AGV	1.63%	CN	0.26%	W+RT	0.05%
W	1.59%	DA	0.25%	W+TV	0.05%
MV	1.56%	FD	0.22%	RP+UP	0.05%
RJ	1.44%	DD	0.18%	W+RY	0.05%
SX	1.43%	TV+RV	0.16%	AG	0.05%
RY	1.29%	S+MP	0.16%	DT	0.05%
RD	1.26%	AS	0.16%	R+RP	0.05%
DJ	1.08%	MJ	0.15%	FN+MP	0.05%
RA	1.04%	IJ	0.15%	AGQ	0.04%
DN	0.99%	RP+MP	0.15%	AGN+S	0.04%
AGN	0.98%	RN+S	0.14%	TV+IV	0.04%
M	0.92%	RV+TV	0.14%	S+UP	0.04%
MC	0.88%	RP+S	0.13%	W+MT	0.04%
UV	0.77%	CQ	0.12%	FN+RP	0.04%
UA	0.68%	X	0.12%	W+UP	0.04%
MN	0.64%	UJ	0.12%	RJ+DJ	0.04%
DY	0.58%	RV+FV	0.10%		
IV	0.57%	FJ	0.10%		

Appendix G

Probabilistic AEG samples

Original text	*There are a lot of things that can have an effect on the environment .*
0-POS-O	***Because*** *there are a lot of things that can* ***to*** *have an effect on the environment .*
0-POS-U	***thee*** *are a lot* ***'s*** *of things that can* ***be reaching*** *an effect* ***with regards to*** *the environment .*
0-POS-S	*There are a lot of* ***something*** *that can have an effect* ***or in forest*** *on* ***tomorrow 's*** *environment .*
1-POS-O	*There* ***have*** *a lot of things that can have* ***oun*** *effect on the environment .*
1-POS-U	*There* ***would be*** *a* ***big namber*** *of things that can have an effect* ***on environment*** *.*
1-POS-S	*There are a lot of* ***ours*** *things that can have an* ***re-cycle*** *effect on the environment* ***stuffs*** *.*
2-POS-O	*There* ***were*** *a lot of things that can have an effect on* ***of*** *the environment .*
2-POS-U	*There are a* ***to*** *of things that can have* ***to*** *an effect on the* ***glabal*** *environment .*
2-POS-S	***A*** *lot of things that can have an effect on the environment .*
0-LEX-O	*There are a lot of things that* ***will*** *have* ***had*** *an effect on the environment .*
0-LEX-U	*There are* ***about*** *lot of things that can have an effect on the environment* ***togheter***
0-LEX-S	*There* ***been*** *a lot of things that can* ***effect in*** *the environment*
1-LEX-O	*There are a lot of things that can have an effect on the* ***environmental*** *.*
1-LEX-U	*There* ***we*** *a* ***losts*** *of* ***many*** *things that can have* ***performed*** *an* ***inflence*** *on* ***channel the enviromental*** *.*
1-LEX-S	***they*** *are* ***olot*** *of things that can* ***gonne*** *have an* ***afect*** *on the* ***surrounding*** *.*
2-LEX-O	*There* ***is*** *a lot of things that can have an effect on the environment* ***?***
2-LEX-U	***That 's*** *a lot of things that can have an effect on the environment .*
2-LEX-S	***Are there*** *a lot of* ***similar*** *things that can have an effect on the* ***surrounding*** *.*

References

Adèr, H. J., Mellenbergh, G. J. and Hand, D. J. (2008). *Advising on Research Methods: A Consultant's Companion*. Huizen, The Netherlands: Johannes Van Kessel.

Andersen, Ø. E., Yannakoudakis, H., Barker, F. and Parish, T. (2013). Developing and testing a self-assessment and tutoring system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. BEA 2013. Atlanta: Association for Computational Linguistics, pp. 32–41.

Antoniadis, G., Granger, S., Kraif, O., Ponton, C. and Zampa, V. (2006). NLP and CALL: integration is working. In *Proceedings of the 7th Conference of Teaching and Language Corpora (TaLC7)*. Paris.

Arppe, A. (2000). Developing a grammar checker for Swedish. In *Proceedings of the Twelfth Nordic Conference in Computational Linguistics (NoDaLiDa)*. Trondheim, Norway, pp. 13–27.

Ashwell, T. (2000). Patterns of Teacher Response to Student Writing in a Multiple-Draft Composition Classroom: Is Content Feedback Followed by Form Feedback the Best Method? In *Journal of Second Language Writing* 9 (3), pp. 227–257.

Attali, Y. (2004). Exploring the feedback and revision features of Criterion. In *Proceedings of the Annual Meeting of the National Council on Measurement in Education*. San Diego, pp. 1–22.

Baroni, M., Bernardini, S., Ferraresi, A. and Zanchetta, E. (2009). The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. In *Language Resources and Evaluation* 43 (3), pp. 209–226.

Bartram, M. and Walton, R. (1991). *Correction: A Positive Approach to Language Mistakes*. Hove, England: Language Teaching Publications.

Behera, B. and Bhattacharyya, P. (2013). Automated Grammar Correction Using Hierarchical Phrase-Based Statistical Machine Translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Nagoya, Japan: Asian Federation of Natural Language Processing, pp. 937–941.

Berend, G., Vincze, V., Zarriess, S. and Farkas, R. (2013). LFG-based Features for Noun Number and Article Grammatical Errors. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 62–67.

Berg-Kirkpatrick, T., Burkett, D. and Klein, D. (2012). An Empirical Investigation of Statistical Significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Jeju Island, Korea: Association for Computational Linguistics, pp. 995–1005.

Bergsma, S., Lin, D. and Goebel, R. (2009). Web-scale n-gram models for lexical disambiguation. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*. IJCAI'09. Pasadena: Morgan Kaufmann Publishers Inc., pp. 1507–1512.

Berzak, Y., Reichart, R. and Katz, B. (2015). Contrastive Analysis with Predictive Power: Typology Driven Estimation of Grammatical Error Distributions in ESL. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*. Beijing, China: Association for Computational Linguistics, pp. 94–102.

Bigert, J. (2004). Probabilistic detection of context-sensitive spelling errors. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-04)*. Vol. 5. 1. Lisbon, Portugal: European Language Resources Association (ELRA), pp. 1633–1636.

Bird, S., Dale, R., Dorr, B. J., Gibson, B., Joseph, M. T., Kan, M.-Y., Lee, D., Powley, B., Radev, D. R. and Tan, Y. F. (2008). The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In pp. 1755–1759.

Bird, S., Loper, E. and Klein, E. (2009). *Natural Language Processing with Python*. O'Reilly Media Inc.

Bitchener, J. and Knoch, U. (2008). The value of written corrective feedback for migrant and international students. In *Language Teaching Research* 12 (3), pp. 409–431.

Bitchener, J. and Knoch, U. (2010). The Contribution of Written Corrective Feedback to Language Development: A Ten Month Investigation. In *Applied Linguistics* 31 (2), pp. 193–214.

Blatz, J., Fitzgerald, E., Foster, G., Gandrabur, S., Goutte, C., Kulesza, A., Sanchis, A. and Ueffing, N. (2004). Confidence Estimation for Machine Translation. In *Proceedings of the 20th International Conference on Computational Linguistics*. COLING '04. Geneva, Switzerland: Association for Computational Linguistics.

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. (2015). Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1–46.

Boroş, T., Dumitrescu, S. D., Zafiu, A., Barbu Mititelu, V. and Vaduva, I. P. (2014). RACAI GEC – A hybrid approach to Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language*

Learning: Shared Task. Baltimore, Maryland: Association for Computational Linguistics, pp. 43–48.

Bosch, A. van den and Berck, P. (2013). Memory-based Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 102–108.

Bouamor, H., Sajjad, H., Durrani, N. and Oflazer, K. (2015). QCMUQ@QALB-2015 Shared Task: Combining Character level MT and Error-tolerant Finite-State Recognition for Arabic Spelling Correction. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*. Beijing, China: Association for Computational Linguistics, pp. 144–149.

Bougares, F. and Bouamor, H. (2015). UMMU@QALB-2015 Shared Task: Character and Word level SMT pipeline for Automatic Error Correction of Arabic Text. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*. Beijing, China: Association for Computational Linguistics, pp. 166–172.

Brants, T. and Franz, A. (2006). *Web 1T 5-gram Version 1 LDC2006T13*. DVD. Philadelphia.

Brants, T. and Franz, A. (2009). *Web 1T 5-gram, 10 European Languages Version 1 LDC2009T25*. Web Download. Philadelphia.

Briscoe, T., Carroll, J. and Watson, R. (2006). The second release of the RASP system. In *Proceedings of the COLING/ACL on Interactive presentation sessions*. COLING-ACL '06. Sydney, Australia: Association for Computational Linguistics, pp. 77–80.

Brockett, C., Dolan, W. B. and Gamon, M. (2006). Correcting ESL Errors Using Phrasal SMT Techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia: Association for Computational Linguistics, pp. 249–256.

Brooke, J. and Hirst, G. (2012a). Measuring Interlanguage: Native Language Identification with L1-influence Metrics. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 2012)*. Istanbul, Turkey: European Language Resources Association (ELRA), pp. 779–784.

Brooke, J. and Hirst, G. (2012b). Robust, Lexicalized Native Language Identification. In *Proceedings of COLING 2012*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 391–408.

Brown, H. D. (2014). *Principles of Language Learning and Teaching*. Always learning. Upper Saddle River, New Jersey: Pearson Education.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D. and Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. In *Comput. Linguist.* 19 (2), pp. 263–311.

Bryant, C. and Ng, H. T. (2015). How Far are We from Fully Automatic High Quality Grammatical Error Correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint*

Conference on Natural Language Processing (Volume 1: Long Papers). Beijing, China: Association for Computational Linguistics, pp. 697–707.

Buckland, M. and Gey, F. (1994). The relationship between Recall and Precision. In *Journal of the American Society for Information Science* 45 (1), pp. 12–19.

Bustamante, F. R. and León, F. S. (1996). GramCheck: A Grammar and Style Checker. In *Proceedings of the 16th International Conference on Computational Linguistics*. Vol. 1. Copenhagen, Denmark, pp. 175–181.

Buys, J. and Merwe, B. van der (2013). A Tree Transducer Model for Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 43–51.

Bykh, S. and Meurers, D. (2014). Exploring Syntactic Features for Native Language Identification: A Variationist Perspective on Feature Encoding and Ensemble Optimization. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 1962–1973.

Bykh, S., Vajjala, S., Krivanek, J. and Meurers, D. (2013). Combining Shallow and Linguistically Motivated Features in Native Language Identification. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Atlanta, Georgia: Association for Computational Linguistics, pp. 197–206.

Byun, J., Rim, H.-C. and Park, S.-Y. (2007). Automatic Spelling Correction Rule Extraction and Application for Spoken-Style Korean Text. In *Proceedings of the Sixth International Conference on Advanced Language Processing and Web Information Technology*. Luoyang, China, pp. 195–199.

Cahill, A., Chodorow, M., Wolff, S. and Madnani, N. (2013a). Detecting Missing Hyphens in Learner Text. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Atlanta, Georgia: Association for Computational Linguistics, pp. 300–305.

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

Carrillo, H. and Lipman, D. (1988). The Multiple Sequence Alignment Problem in Biology. In *SIAM Journal of Applied Mathematics* 48 (5), pp. 1073–1082.

Chang, T., Sung, Y., Hong, J. and Chang, J. (2014). KNGED: A tool for grammatical error diagnosis of Chinese sentences. In *Workshop Proceedings of the 22nd International Conference on Computers in Education (ICCE 2014)*. Nara, Japan: Asia-Pacific Society for Computers in Education, pp. 48–55.

Chen, S. F. and Goodman, J. (1998). *An Empirical Study of Smoothing Techniques for Language Modeling*. Technical report TR-10-98. Harvard University.

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

Chodorow, M., Gamon, M. and Tetreault, J. (2010). The utility of article and preposition error correction systems for English language learners: Feedback and assessment. In *Language Testing* 27 (3), pp. 419–436.

Chodorow, M. and Leacock, C. (2000). An unsupervised method for detecting grammatical errors. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics (NAACL)*. Seattle: Association for Computational Linguistics, pp. 140–147.

Choi, J. and Lee, Y. (2010). The Use of Feedback in the ESL Writing Class Integrating Automated Essay Scoring. In *Proceedings of Society for Information Technology & Teacher Education International Conference*. Ed. by D. Gibson and B. Dodge. San Diego: Association for the Advancement of Computing in Education, pp. 3008–3012.

Cohen, A. D. and Robbins, M. (1976). Toward assessing interlanguage performance: The relationship between selected errors, learners' characteristics, and learners' expectations. In *Language Learning* 26 (1), pp. 45–66.

Corder, S. P. (1967). The significance of learner's errors. In *IRAL-International Review of Applied Linguistics in Language Teaching* 5 (1-4), pp. 161–170.

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

Crystal, D. (2008). Two thousand million? In *English Today* 24 (1), pp. 3–6.

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

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

Dahlmeier, D. and Ng, H. T. (2012b). Better Evaluation for Grammatical Error Correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. NAACL 2012. Montreal, Canada, pp. 568–572.

Dahlmeier, D., Ng, H. T. and Ng, E. J. F. (2012). NUS at the HOO 2012 Shared Task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Montréal, Canada: Association for Computational Linguistics, pp. 216–224.

Dahlmeier, D., Ng, H. T. and Tran, T. P. (2011). NUS at the HOO 2011 Pilot Shared Task. In *Proceedings of the Generation Challenges Session at the 13th*

European Workshop on Natural Language Generation. Nancy, France: Association for Computational Linguistics, pp. 257–259.

Dahlmeier, D., Ng, H. T. and Wu, S. M. (2013). Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Atlanta, Georgia: Association for Computational Linguistics, pp. 22–31.

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

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

Dale, R. and Narroway, G. (2011). *The HOO Pilot Data Set: Notes on Release 2.0*.

Daudaravicius, V., Banchs, R. E., Volodina, E. and Napoles, C. (2016). 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*. San Diego, CA: Association for Computational Linguistics, pp. 53–62.

De Felice, R. and Pulman, S. G. (2008). A Classifier-Based Approach to Preposition and Determiner Error Correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*. Manchester, UK: Coling 2008 Organizing Committee, pp. 169–176.

Dickinson, M. (2010). Generating Learner-Like Morphological Errors in Russian. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*. Beijing, China: Coling 2010 Organizing Committee, pp. 259–267.

Domeij, R., Knutsson, O., Carlberger, J. and Kann, V. (2000). Granska - an efficient hybrid system for Swedish grammar checking. In *Proceedings of the 12th Nordic Conference in Computational Linguistics (Nodalida-99)*. Trondheim, Norway: Department of Linguistics, University of Trondheim, pp. 28–40.

Efron, B. and Tibshirani, R. (1993). *An Introduction to the Bootstrap*. New York: Chapman & Hall.

Ehsan, N. and Faili, H. (2010). Towards grammar checker development for Persian language. In *6th IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLPKE'10)*. Beijing, China, pp. 150–157.

Ehsan, N. and Faili, H. (2013). Grammatical and context-sensitive error correction using a statistical machine translation framework. In *Software: Practice and Experience* 43 (2), pp. 187–206.

Elghafari, A., Meurers, D. and Wunsch, H. (2010). Exploring the Data-Driven Prediction of Prepositions in English. In *Coling 2010: Posters*. Beijing, China: Coling 2010 Organizing Committee, pp. 267–275.

Ellis, R. (2008). *The Study of Second Language Acquisition*. Oxford applied linguistics. Oxford: Oxford University Press.

Ellis, R., Sheen, Y., Murakami, M. and Takashima, H. (2008). The effects of focused and unfocused written corrective feedback in an English as a foreign language context. In *System* 36 (3), pp. 353–371.

Fallman, D. (2002). The Penguin: Using the Web As a Database for Descriptive and Dynamic Grammar and Spell Checking. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '02. Minneapolis: ACM, pp. 616–617.

Federico, M., Bertoldi, N. and Cettolo, M. (2008). IRSTLM: an open source toolkit for handling large scale language models. In *Proceedings of the 9th Annual Conference of the International Speech Communication Association*. INTERSPEECH 2008. Brisbane, Australia: ISCA, pp. 1618–1621.

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

Felice, M. and Yuan, Z. (2014a). 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*. Gothenburg, Sweden: Association for Computational Linguistics, pp. 116–126.

Felice, M. and Yuan, Z. (2014b). To Err is Human, to Correct is Divine. In *XRDS* 21 (1), pp. 22–27.

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

Ferris, D. R. (2006). Does error feedback help student writers? New evidence on the short- and long-term effects of written error correction. In *Feedback in second language writing: Contexts and issues*. Ed. by K. Hyland and F. Hyland. Cambridge: Cambridge University Press, pp. 81–104.

Ferris, D. R. (2011). *Treatment of error in second language student writing*. 2nd ed. The Michigan series on teaching multilingual writers. Ann Arbor: University of Michigan Press.

Ferris, D. R. and Hedgcock, J. S. (2014). *Teaching L2 Composition: Purpose, Process, and Practice*. 3rd ed. New York: Routledge.

Flickinger, D. and Yu, J. (2013). Toward More Precision in Correction of Grammatical Errors. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 68–73.

Fossati, D. and Di Eugenio, B. (2008). I saw TREE trees in the park: How to Correct Real-Word Spelling Mistakes. In *Proceedings of the Sixth International*

Conference on Language Resources and Evaluation (LREC'08). Ed. by N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis and D. Tapias. Marrakech, Morocco: European Language Resources Association (ELRA), pp. 896–901.

Foster, G. and Kuhn, R. (2012). *Forced Decoding for Phrase Extraction*. Technical report.

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

Francis, W. N. and Kucera, H. (1979). *The Brown Corpus: A Standard Corpus of Present-Day Edited American English*. Brown University Liguistics Department. Providence, Rhode Island.

Gale, W. A., Church, K. W. and Yarowsky, D. (1992). One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*. HLT '91. Harriman, New York: Association for Computational Linguistics, pp. 233–237.

Gamon, M. (2010). Using Mostly Native Data to Correct Errors in Learners' Writing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, pp. 163–171.

Gamon, M. and Leacock, C. (2010). Search right and thou shalt find ... Using Web Queries for Learner Error Detection. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*. Los Angeles, California: Association for Computational Linguistics, pp. 37–44.

Gamon, M., Leacock, C., Brockett, C., Dolan, W. B., Gao, J., Belenko, D. and Klementiev, A. (2009). Using Statistical Techniques and Web Search to Correct ESL Errors. In *CALICO Journal* 26 (3), pp. 491–511.

Gass, S. and Selinker, L. (1992). *Language Transfer in Language Learning*. Language acquisition & language disorders. Philadelphia: John Benjamins Publishing Company.

Gass, S. and Selinker, L. (1994). *Second language acquisition: an introductory course*. Topics in applied psycholinguistics. Hillsdale, New Jersey: L. Erlbaum Associates.

Gavrila, M. and Vertan, C. (2011). Training Data in Statistical Machine Translation - the More, the Better? In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*. Hissar, Bulgaria: RANLP 2011 Organising Committee, pp. 551–556.

Geertzen, J., Alexopoulou, T. and Korhonen, A. (2012). Automatic Linguistic Annotation of Large Scale L2 Databases: The EF-Cambridge Open Language Database (EFCamDat). In *Proceedings of the 31st Second Language Research Forum (SLRF)*. Pittsburgh: Cascadilla Proceedings Project, pp. 240–254.

Gillard, P. and Gadsby, A. (1998). Using a learners' corpus in compiling ELT dictionaries. In *Learner English on computer*. Ed. by S. Granger. Studies in language and linguistics. New York: Longman, pp. 159–171.

Granger, S. (2003a). Error-tagged Learner Corpora and CALL: A Promising Synergy. In *CALICO Journal* 20 (3), pp. 465–480.

Granger, S. (2003b). The International Corpus of Learner English: A New Resource for Foreign Language Learning and Teaching and Second Language Acquisition Research. In *TESOL Quarterly* 37 (3), pp. 538–546.

Grundkiewicz, R. and Junczys-Dowmunt, M. (2014). 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*. Ed. by A. Przepiórkowski and M. Ogrodniczuk. Vol. 8686. Springer, pp. 478–490.

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

Gupta, A. (2014). Grammatical Error Detection Using Tagger Disagreement. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 49–52.

Gusfield, D. (1997). *Algorithms on Strings, Trees and Sequences*. Cambridge University Press.

Han, N.-R., Chodorow, M. and Leacock, C. (2006). Detecting Errors in English Article Usage by Non-native Speakers. In *Journal of Natural Language Engineering* 12 (2), pp. 115–129.

Hawkey, R. and Milanovic, M. (2013). *Cambridge English Exams - The First Hundred Years: A History of English Language Assessment from the University of Cambridge, 1913-2013*. Studies in Language Testing. Cambridge: Cambridge University Press.

Heidorn, G., Jensen, K., Miller, L., Byrd, R. and Chodorow, M. (1982). The EPISTLE text-critiquing system. In *IBM Systems Journal* 21 (3), pp. 305–326.

Heilman, M., Cahill, A., Madnani, N., Lopez, M., Mulholland, M. and Tetreault, J. (2014). Predicting Grammaticality on an Ordinal Scale. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, pp. 174–180.

Hermet, M. and Désilets, A. (2009). Using first and second language models to correct preposition errors in second language authoring. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*. EdAppsNLP '09. Boulder, Colorado: Association for Computational Linguistics, pp. 64–72.

Hermet, M., Désilets, A. and Szpakowicz, S. (2008). Using the Web as a Linguistic Resource to Automatically Correct Lexico-Syntactic Errors. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. Ed. by N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S.

Piperidis and D. Tapias. Marrakech, Morocco: European Language Resources Association (ELRA), pp. 874–878.

Hernandez, S. D. and Calvo, H. (2014). CoNLL 2014 Shared Task: Grammatical Error Correction with a Syntactic N-gram Language Model from a Big Corpora. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 53–59.

Herron, D., Menzel, W., Atwell, E., Bisiani, R., Daneluzzi, F., Morton, R., Schmidt, J. A. and Verlag, E. K. (1999). Automatic Localization And Diagnosis Of Pronunciation Errors For Second-Language Learners Of English. In *Proceedings of the Sixth European Conference on Speech Communication and Technology (EUROSPEECH 1999)*. Budapest, Hungary: ISCA, pp. 896–901.

Howson, P. (2013). *The English Effect*. `http://www.britishcouncil.org/sites/default/files/english-effect-report-v2.pdf` [Online; accessed 22 October 2015]. British Council.

Imamura, K., Saito, K., Sadamitsu, K. and Nishikawa, H. (2012). Grammar Error Correction Using Pseudo-Error Sentences and Domain Adaptation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju Island, Korea: Association for Computational Linguistics, pp. 388–392.

Islam, A. and Inkpen, D. (2011). Correcting Different Types of Errors in Texts. In *Proceedings of the 24th Canadian Conference on Advances in Artificial Intelligence*. Canadian AI'11. St. John's, Canada: Springer-Verlag, pp. 192–203.

Izumi, E., Uchimoto, K. and Isahara, H. (2004). SST speech corpus of Japanese learners' English and automatic detection of learners' errors. In *International Computer Archive of Modern and Medieval English (ICAME)* (28), pp. 31–48.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T. and Isahara, H. (2003). Automatic Error Detection in the Japanese Learners' English Spoken Data. In *The Companion Volume to the Proceedings of 41st Annual Meeting of the Association for Computational Linguistics*. Sapporo, Japan: Association for Computational Linguistics, pp. 145–148.

James, C. (1998). *Errors in Language Learning and Use: Exploring Error Analysis*. Applied linguistics and language study. London: Longman.

Jeblee, S., Bouamor, H., Zaghouani, W. and Oflazer, K. (2014). CMUQ@QALB-2014: An SMT-based System for Automatic Arabic Error Correction. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 137–142.

Jia, Z., Wang, P. and Zhao, H. (2013). Grammatical Error Correction as Multiclass Classification with Single Model. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 74–81.

Johannessen, J. B., Hagen, K. and Lane, P. (2002). The Performance of a Grammar Checker with Deviant Language Input. In *Proceedings of the 19th*

International Conference on Computational Linguistics - Volume 2. COLING '02. Taipei, Taiwan: Association for Computational Linguistics, pp. 1–8.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2014). The AMU System in the CoNLL-2014 Shared Task: Grammatical Error Correction by Data-Intensive and Feature-Rich Statistical Machine Translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 25–33.

Kanters, R., Cucchiarini, C. and Strik, H. (2009). The Goodness of Pronunciation Algorithm : a Detailed Performance Study. In *In Proceedings of the ISCA Workshop on Speech and Language Technology in Education (SLaTE 2009)*. Warwickshire, England: ISCA, pp. 2–5.

Kao, T.-H., Chang, Y.-W., Chiu, H.-W., Yen, T.-H., Boisson, J., Wu, J.-C. and Chang, J. S. (2013). CoNLL-2013 Shared Task: Grammatical Error Correction NTHU System Description. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 20–25.

Kilgarriff, A. (2007). Googleology is Bad Science. In *Computational Linguistics* 33 (1), pp. 147–151.

Kneser, R. and Ney, H. (1995). Improved backing-off for M-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Vol. 1. Detroit, Michigan: IEEE Signal Processing Society, pp. 181–184.

Kochmar, E. (2011). Identification of a Writer's Native Language by Error Analysis. Master's thesis. Cambridge: University of Cambridge.

Kochmar, E. and Briscoe, T. (2013). Capturing Anomalies in the Choice of Content Words in Compositional Distributional Semantic Space. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*. Hissar, Bulgaria: INCOMA Ltd., pp. 365–372.

Koehn, P. (2004). Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of EMNLP 2004*. Ed. by D. Lin and D. Wu. Barcelona, Spain: Association for Computational Linguistics, pp. 388–395.

Koehn, P. (2010). *Statistical Machine Translation*. 1st ed. New York: Cambridge University Press.

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. (2007). Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*. ACL '07. Prague, Czech Republic: Association for Computational Linguistics, pp. 177–180.

Koehn, P., Och, F. J. and Marcu, D. (2003). 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 - Volume 1*. NAACL '03. Edmonton, Canada: Association for Computational Linguistics, pp. 48–54.

Krashen, S. (1982). *Principles and practice in second language acquisition*. Language teaching methodology series. Oxford: Pergamon.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. In *The Annals of Mathematical Statistics* 22 (1), pp. 79–86.

Kunchukuttan, A., Chaudhury, S. and Bhattacharyya, P. (2014). Tuning a Grammar Correction System for Increased Precision. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 60–64.

Kunchukuttan, A., Shah, R. and Bhattacharyya, P. (2013). IITB System for CoNLL 2013 Shared Task: A Hybrid Approach to Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 82–87.

Lado, R. (1957). *Linguistics across cultures: applied linguistics for language teachers*. Ann Arbor: University of Michigan Press.

Lavolette, E., Polio, C. and Kahng, J. (2015). The Accuracy of Computer-Assisted Feedback and Students' Responses to It. In *Language Learning & Technology* 19 (2), pp. 50–68.

Leacock, C. and Chodorow, M. (2003). Automated grammatical error detection. In *Automated Essay Scoring: A Cross-Disciplinary Perspective*. Ed. by M. Shermis and J. Burstein. Mahwah, NJ: Lawrence Erlbaum Associates, pp. 195–207.

Leacock, C., Chodorow, M., Gamon, M. and Tetreault, J. (2014). *Automated Grammatical Error Detection for Language Learners*. 2nd ed. Synthesis Lectures on Human Language Technologies. San Rafael, California: Morgan & Claypool Publishers.

Lee, J. (2004). Automatic Article Restoration. In *HLT-NAACL 2004: Student Research Workshop*. Ed. by D. M. Susan Dumais and S. Roukos. Boston, Massachusetts, USA: Association for Computational Linguistics, pp. 31–36.

Lee, J. and Seneff, S. (2008). Correcting Misuse of Verb Forms. In *Proceedings of ACL-08: HLT*. Columbus, Ohio: Association for Computational Linguistics, pp. 174–182.

Lee, K. and Lee, G. G. (2014). POSTECH Grammatical Error Correction System in the CoNLL-2014 Shared Task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 65–73.

Lee, L.-H., Yu, L.-C. and Chang, L.-P. (2015). Overview of the NLP-TEA 2015 Shared Task for Chinese Grammatical Error Diagnosis. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*. Beijing, China: Association for Computational Linguistics, pp. 1–6.

Lee, L.-H., Yu, L.-C., Lee, K.-C., Tseng, Y.-H., Chang, L.-P. and Chen, H.-H. (2014). A Sentence Judgment System for Grammatical Error Detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: System Demonstrations*. Dublin, Ireland: Dublin City University and Association for Computational Linguistics, pp. 67–70.

Lewis, M. P., Simons, G. F. and Fennig, C. D., eds. (2015). *Ethnologue: Languages of the World*. 18th ed. Online version: `http://www.ethnologue.com`. Dallas, Texas: SIL International.

Lin, C.-J. and Chen, S.-H. (2015). NTOU Chinese Grammar Checker for CGED Shared Task. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*. Beijing, China: Association for Computational Linguistics, pp. 15–19.

Lin, J. (1991). Divergence Measures Based on the Shannon Entropy. In *IEEE Transactions on Information Theory* 37 (1), pp. 145–151.

Lipnevich, A. A. and Smith, J. K. (2008). *Response to assessment feedback: The effects of differential feedback on students' performance*. Technical report. Educational Testing Service.

Macdonald, N., Frase, L., Gingrich, P. and Keenan, S. (1982). The Writer's Workbench: Computer Aids for Text Analysis. In *Communications, IEEE Transactions on* 30 (1), pp. 105–110.

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

Madnani, N., Tetreault, J. and Chodorow, M. (2012). Exploring grammatical error correction with not-so-crummy machine translation. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Montreal, Canada: Association for Computational Linguistics, pp. 44–53.

Mangu, L. and Brill, E. (1997). Automatic Rule Acquisition for Spelling Correction. In *Proceedings of the Fourteenth International Conference on Machine Learning*. ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 187–194.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press.

Miller, G. A. (1995). WordNet: a lexical database for English. In *Communications of the ACM* 38 (11), pp. 39–41.

Minnen, G., Bond, F. and Copestake, A. (2000). Memory-based learning for article generation. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*. Lisbon, Portugal: Association for Computational Linguistics, pp. 43–48.

Mizumoto, T., Hayashibe, Y., Komachi, M., Nagata, M. and Matsumoto, Y. (2012). The Effect of Learner Corpus Size in Grammatical Error Correction of ESL Writings. In *Proceedings of COLING 2012: Posters*. Mumbai, India: The COLING 2012 Organizing Committee, pp. 863–872.

Mizumoto, T., Komachi, M., Nagata, M. and Matsumoto, Y. (2011). Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, pp. 147–155.

Mohit, B., Rozovskaya, A., Habash, N., Zaghouani, W. and Obeid, O. (2014). The First QALB Shared Task on Automatic Text Correction for Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 39–47.

Mostefa, D., Abualasal, J., Asbayou, O., Gzawi, M. and Abbès, R. (2015). TECHLIMED@QALB-Shared Task 2015: a hybrid Arabic Error Correction System. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*. Beijing, China: Association for Computational Linguistics, pp. 161–165.

Mostefa, D., Asbayou, O. and Abbes, R. (2014). TECHLIMED system description for the Shared Task on Automatic Arabic Error Correction. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 155–159.

Mount, D. W. (2004). *Bioinformatics: Sequence and Genome Analysis*. 2nd ed. Cold Spring Harbor Laboratory Press.

Naber, D. (2003). A Rule-Based Style and Grammar Checker. Bachelor's thesis. Bielefeld, Germany: Universität Bielefeld.

Nagata, R. and Nakatani, K. (2010). Evaluating performance of grammatical error detection to maximize learning effect. In *Coling 2010: Posters*. Beijing, China: Coling 2010 Organizing Committee, pp. 894–900.

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

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H. and Bryant, C. (2014). The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 1–14.

Ng, H. T., Wu, S. M., Wu, Y., Hadiwinoto, C. and Tetreault, J. (2013). The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1–12.

Nicholls, D. (2003). The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*. Ed. by D. Archer, P. Rayson, A. Wilson and T. McEnery. Lancaster, UK: University Centre for Computer Corpus Research on Language, Lancaster University, pp. 572–581.

Och, F. J. and Ney, H. (2003). A systematic comparison of various statistical alignment models. In *Comput. Linguist.* 29 (1), pp. 19–51.

Okanohara, D. and Tsujii, J. (2007). A discriminative language model with pseudo-negative samples. In *Proceedings of the 45th Annual Meeting of the*

Association of Computational Linguistics. Prague, Czech Republic: Association for Computational Linguistics, pp. 73–80.

Papineni, K., Roukos, S., Ward, T. and Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318.

Park, B.-R. and Rim, H.-C. (1998). Recognizing Unknown Words and Correcting Spelling errors as Preprocessing for Korean Information Processing System. In *Transactions of the Korea Information Processing Society* 5 (10), pp. 2591–2599.

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

Parker, R., Graff, D., Kong, J., Chen, K. and Maeda, K. (2011). *English Gigaword Fifth Edition LDC2011T07*. DVD. Philadelphia.

Pedersen, T. and Kolhatkar, V. (2009). WordNet::SenseRelate::AllWords: a broad coverage word sense tagger that maximizes semantic relatedness. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Demonstration Session*. NAACL-Demonstrations '09. Boulder, Colorado: Association for Computational Linguistics, pp. 17–20.

Polio, C., Fleck, C. and Leder, N. (1998). "If I only had more time:" ESL learners' changes in linguistic accuracy on essay revisions. In *Journal of Second Language Writing* 7 (1), pp. 43–68.

Pravec, N. A. (2002). Survey of learner corpora. In *International Computer Archive of Modern and Medieval English (ICAME)* 26 (1), pp. 81–114.

Prodromou, L. (2008). *English as a Lingua Franca: A Corpus-based Analysis*. London: Continuum.

Prokofyev, R., Mavlyutov, R., Grund, M., Demartini, G. and Cudré-Mauroux, P. (2014). Correct Me If I'M Wrong: Fixing Grammatical Errors by Preposition Ranking. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. CIKM '14. Shanghai, China: ACM, pp. 331–340.

Putra, D. D. and Szabo, L. (2013). UdS at CoNLL 2013 Shared Task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 88–95.

Quan, L., Kolomiyets, O. and Moens, M.-F. (2012). KU Leuven at HOO-2012: A Hybrid Approach to Detection and Correction of Determiner and Preposition Errors in Non-native English Text. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Montréal, Canada: Association for Computational Linguistics, pp. 263–271.

Richardson, S. D. and Braden-Harder, L. C. (1988). The Experience of Developing a Large-scale Natural Language Text Processing System: CRITIQUE.

In *Proceedings of the Second Conference on Applied Natural Language Processing*. ANLC '88. Austin, Texas: Association for Computational Linguistics, pp. 195–202.

Rozovskaya, A., Bouamor, H., Habash, N., Zaghouani, W., Obeid, O. and Mohit, B. (2015). The Second QALB Shared Task on Automatic Text Correction for Arabic. In *Proceedings of the Second Workshop on Arabic Natural Language Processing*. Beijing, China: Association for Computational Linguistics, pp. 26–35.

Rozovskaya, A., Chang, K.-W., Sammons, M. and Roth, D. (2013). The University of Illinois System in the CoNLL-2013 Shared Task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 13–19.

Rozovskaya, A., Chang, K.-W., Sammons, M., Roth, D. and Habash, N. (2014a). The Illinois-Columbia System in the CoNLL-2014 Shared Task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 34–42.

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

Rozovskaya, A. and Roth, D. (2010b). Generating Confusion Sets for Context-Sensitive Error Correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Cambridge, MA: Association for Computational Linguistics, pp. 961–970.

Rozovskaya, A. and Roth, D. (2010c). Training paradigms for correcting errors in grammar and usage. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. HLT '10. Los Angeles: Association for Computational Linguistics, pp. 154–162.

Rozovskaya, A. and Roth, D. (2011). Algorithm Selection and Model Adaptation for ESL Correction Tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon: Association for Computational Linguistics, pp. 924–933.

Rozovskaya, A. and Roth, D. (2013). Joint Learning and Inference for Grammatical Error Correction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, pp. 791–802.

Rozovskaya, A., Roth, D. and Srikumar, V. (2014b). Correcting Grammatical Verb Errors. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Gothenburg, Sweden: Association for Computational Linguistics, pp. 358–367.

Rozovskaya, A., Sammons, M., Gioja, J. and Roth, D. (2011). University of Illinois System in HOO Text Correction Shared Task. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*. Nancy, France: Association for Computational Linguistics, pp. 263–266.

Rozovskaya, A., Sammons, M. and Roth, D. (2012). The UI System in the HOO 2012 Shared Task on Error Correction. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Montréal, Canada: Association for Computational Linguistics, pp. 272–280.

Sakaguchi, K., Arase, Y. and Komachi, M. (2013). Discriminative Approach to Fill-in-the-Blank Quiz Generation for Language Learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 238–242.

Sakaguchi, K., Hayashibe, Y., Kondo, S., Kanashiro, L., Mizumoto, T., Komachi, M. and Matsumoto, Y. (2012). NAIST at the HOO 2012 Shared Task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Montréal, Canada: Association for Computational Linguistics, pp. 281–288.

Sawai, Y., Komachi, M. and Matsumoto, Y. (2013). A Learner Corpus-based Approach to Verb Suggestion for ESL. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 708–713.

Schiftner, B. (2008). Learner Corpora of English and German: What is their status quo and where are they headed. In *Vienna English Working Papers (VIEWS)* 17 (2), pp. 47–78.

Seo, H., Lee, J., Kim, S., Lee, K., Kang, S. and Lee, G. G. (2012). A Meta Learning Approach to Grammatical Error Correction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Jeju Island, Korea: Association for Computational Linguistics, pp. 328–332.

Shannon, C. E. (1948). A Mathematical Theory of Communication. In *The Bell System Technical Journal* 27 (3), pp. 379–423.

Sheen, Y. (2007). The Effect of Focused Written Corrective Feedback and Language Aptitude on ESL Learners' Acquisition of Articles. In *TESOL Quarterly* 41 (2), pp. 255–283.

Shermis, M. D., Garvan, C. W. and Diao, Y. (2008). The Impact of Automated Essay Scoring on Writing Outcomes. In *Proceedings of the Annual Meeting of the National Council on Measurement in Education*. New York, pp. 1–45.

Sidorov, G., Gupta, A., Tozer, M., Català, D., Catena, A. and Fuentes, S. (2013). Rule-based System for Automatic Grammar Correction Using Syntactic N-grams for English Language Learning (L2). In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 96–101.

Sjöbergh, J. and Knutsson, O. (2005). Faking Errors to avoid Making Errors: Machine Learning for Error Detection in Writing. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP)*. Borovets, Bulgaria, pp. 506–512.

Smucker, M. D., Allan, J. and Carterette, B. (2007). A Comparison of Statistical Significance Tests for Information Retrieval Evaluation. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*. CIKM '07. Lisbon, Portugal: ACM, pp. 623–632.

Specia, L., Turchi, M., Cancedda, N., Dymetman, M. and Cristianini, N. (2009). Estimating the Sentence-Level Quality of Machine Translation Systems. In *13th Annual Conference of the European Association for Machine Translation*. EAMT. Barcelona, Spain, pp. 28–37.

Sun, C., Jin, X., Lin, L., Zhao, Y. and Wang, X. (2015). Convolutional Neural Networks for Correcting English Article Errors. In *Natural Language Processing and Chinese Computing*. Ed. by J. Li, H. Ji, D. Zhao and Y. Feng. Vol. 9362. Lecture Notes in Computer Science. Springer International Publishing, pp. 102–110.

Suresh, B. (2010). *Inclusion of large input corpora in Statistical Machine Translation*. Technical report. Stanford University.

Swan, M. and Smith, B. (2001). *Learner English: A Teacher's Guide to Interference and Other Problems*. Cambridge Handbooks for Language Teachers. Cambridge: Cambridge University Press.

Tetreault, J., Blanchard, D. and Cahill, A. (2013). A Report on the First Native Language Identification Shared Task. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Atlanta, Georgia: Association for Computational Linguistics, pp. 48–57.

Tetreault, J. and Chodorow, M. (2009). Examining the use of region web counts for ESL error detection. In *Proceedings of the Web as Corpus Workshop (WAC-5)*. San Sebastian, Spain: Elhuyar Fundazioa.

Tetreault, J., Foster, J. and Chodorow, M. (2010). Using Parse Features for Preposition Selection and Error Detection. In *Proceedings of the ACL 2010 Conference Short Papers*. Uppsala, Sweden: Association for Computational Linguistics, pp. 353–358.

Tomeh, N., Habash, N., Eskander, R. and Le Roux, J. (2014). A Pipeline Approach to Supervised Error Correction for the QALB-2014 Shared Task. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 114–120.

Tono, Y. (2003). Learner corpora: design, development and applications. In *Proceedings of the 2003 Corpus Linguistics Conference*. Lancaster, UK: UCREL, Lancaster University, pp. 800–809.

Truscott, J. (1996). The Case Against Grammar Correction in L2 Writing Classes. In *Language Learning* 46 (2), pp. 327–369.

Truscott, J. and Hsu, A. Y.-P. (2008). Error correction, revision, and learning. In *Journal of Second Language Writing* 17 (4), pp. 292–305.

van Beuningen, C. G., De Jong, N. H. and Kuiken, F. (2012). Evidence on the Effectiveness of Comprehensive Error Correction in Second Language Writing. In *Language Learning* 62 (1), pp. 1–41.

Vogel, S., Ney, H. and Tillmann, C. (1996). HMM-based Word Alignment in Statistical Translation. In *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*. COLING '96. Copenhagen, Denmark: Association for Computational Linguistics, pp. 836–841.

Wagner, J. (2012). Detecting Grammatical Errors with Treebank-Induced, Probabilistic Parsers. PhD thesis. Dublin, Ireland: Dublin City University.

Wagner, J., Foster, J. and Genabith, J. van (2007). A Comparative Evaluation of Deep and Shallow Approaches to the Automatic Detection of Common Grammatical Errors. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. Prague, Czech Republic: Association for Computational Linguistics, pp. 112–121.

Wagner, J., Foster, J. and Genabith, J. van (2009). Judging Grammaticality: Experiments in Sentence Classification. In *CALICO Journal* 26 (3), pp. 474–490.

Wang, P., Jia, Z. and Zhao, H. (2014a). Grammatical Error Detection and Correction using a Single Maximum Entropy Model. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 74–82.

Wang, Y., Wang, L., Zeng, X., Wong, D. F., Chao, L. S. and Lu, Y. (2014b). Factored Statistical Machine Translation for Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 83–90.

West, R., Park, Y. A. and Levy, R. (2011). Bilingual Random Walk Models for Automated Grammar Correction of ESL Author-Produced Text. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*. Portland, Oregon: Association for Computational Linguistics, pp. 170–179.

Wikipedia (2015). *Wikipedia:Simple English Wikipedia*. `https://en.wikipedia.org/wiki/Wikipedia:Simple_English_Wikipedia` [Online; accessed 10 December 2015].

Wilcox-O'Hearn, A., Hirst, G. and Budanitsky, A. (2008). Real-Word Spelling Correction with Trigrams: A Reconsideration of the Mays, Damerau, and Mercer Model. In *Computational Linguistics and Intelligent Text Processing*. Ed. by A. Gelbukh. Vol. 4919. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 605–616.

Wilcox-O'Hearn, L. A. (2013). A Noisy Channel Model Framework for Grammatical Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 109–114.

Wilcox-O'Hearn, L. A. (2014). Detection is the central problem in real-word spelling correction. In arXiv: `1408.3153` (cs.CL).

Wu, J.-C., Chang, Y.-C., Mitamura, T. and Chang, J. S. (2010). Automatic Collocation Suggestion in Academic Writing. In *Proceedings of the ACL 2010*

Conference Short Papers. Uppsala, Sweden: Association for Computational Linguistics, pp. 115–119.

Wu, J.-C., Yen, T.-H., Chang, J., Huang, G.-C., Chang, J., Hsu, H.-L., Chang, Y.-W. and Chang, J. S. (2014). NTHU at the CoNLL-2014 Shared Task. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 91–95.

Wu, Y. and Ng, H. T. (2013). Grammatical Error Correction Using Integer Linear Programming. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 1456–1465.

Xiang, Y., Yuan, B., Zhang, Y., Wang, X., Zheng, W. and Wei, C. (2013). A Hybrid Model For Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 115–122.

Xie, W., Huang, P., Zhang, X., Hong, K., Huang, Q., Chen, B. and Huang, L. (2015). Chinese Spelling Check System Based on N-gram Model. In *Proceedings of the Eighth SIGHAN Workshop on Chinese Language Processing*. Beijing, China: Association for Computational Linguistics, pp. 128–136.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D. and Ng, A. Y. (2016). Neural Language Correction with Character-Based Attention. In arXiv: 1603.09727 (cs.CL).

Yannakoudakis, H., Briscoe, T. and Medlock, B. (2011). A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, pp. 180–189.

Yi, B.-J., Lee, H.-C. and Rim, H.-C. (2013). KUNLP Grammatical Error Correction System For CoNLL-2013 Shared Task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 123–127.

Yi, X., Gao, J. and Dolan, W. B. (2008). A web-based English proofing system for English as a second language users. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*. IJCNLP. Hyderabad, India: Asian Federation of Natural Language Processing, pp. 619–624.

Yoshimoto, I., Kose, T., Mitsuzawa, K., Sakaguchi, K., Mizumoto, T., Hayashibe, Y., Komachi, M. and Matsumoto, Y. (2013). NAIST at 2013 CoNLL Grammatical Error Correction Shared Task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 26–33.

Yu, L.-C., Lee, L.-H. and Chang, L.-P. (2014). Overview of Grammatical Error Diagnosis for Learning Chinese as a Foreign Language. In *Workshop Proceedings of the 22nd International Conference on Computers in Education (ICCE 2014)*. Nara, Japan: Asia-Pacific Society for Computers in Education, pp. 42–47.

Yuan, Z., Briscoe, T. and Felice, M. (2016). Candidate re-ranking for SMT-based grammatical error correction. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. San Diego, CA: Association for Computational Linguistics, pp. 256–266.

Yuan, Z. and Felice, M. (2013). Constrained Grammatical Error Correction using Statistical Machine Translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*. Sofia, Bulgaria: Association for Computational Linguistics, pp. 52–61.

Zhang, L. and Wang, H. (2014). A Unified Framework for Grammar Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*. Baltimore, Maryland: Association for Computational Linguistics, pp. 96–102.

Zhao, T., Hoshino, A., Suzuki, M., Minematsu, N. and Hirose, K. (2012). Automatic Chinese pronunciation error detection using SVM trained with structural features. In *Spoken Language Technology Workshop (SLT 2012)*. Miami, Florida, USA: IEEE, pp. 473–478.

Zhao, Y., Komachi, M. and Ishikawa, H. (2014). Extracting a Chinese Learner Corpus from the Web: Grammatical Error Correction for Learning Chinese as a Foreign Language with Statistical Machine Translation. In *Workshop Proceedings of the 22nd International Conference on Computers in Education (ICCE 2014)*. Nara, Japan: Asia-Pacific Society for Computers in Education, pp. 56–61.

Zhao, Y., Komachi, M. and Ishikawa, H. (2015). Improving Chinese Grammatical Error Correction with Corpus Augmentation and Hierarchical Phrase-based Statistical Machine Translation. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*. Beijing, China: Association for Computational Linguistics, pp. 111–116.