Adam Mickiewicz University in Poznań
Faculty of Mathematics and Computer Science

Algorithms for Automatic Grammatical Error Correction

Algorytmy automatycznej poprawy błędów językowych

Roman Grundkiewicz

Supervisor:
Dr hab. Krzysztof Jassem
Auxiliary supervisor:
Dr Marcin Junczys-Dowmunt

A thesis for the degree of Doctor of Philosophy in Mathematics
in the field of Information Science
Praca doktorska na stopień doktora nauk matematycznych
w zakresie informatyki

Poznań, 2017

Abstract

This thesis explores the problem of automated grammatical error correction (GEC) in texts written by non-native English speakers. Our main focus is the machine translation approach to GEC. To overcome the data sparsity problem, we have developed a method for the automatic extraction of potential errors from Wikipedia text edition histories, and created the largest publicly available error annotated corpus so far. We investigate the usefulness of automatic GEC-specific metrics on the basis of their correlation with human judgements by conducting the first large-scale human evaluation study of automated GEC systems. Our proposed phrase-based statistical machine translation (SMT) system achieved new state-of-the-art results on the CoNLL-2014 test data — a standard benchmark for GEC provided during the Conference on Natural Language Learning shared task in 2014. We have shown that parameter optimization towards the task-specific evaluation metric and new GEC-adapted dense features are crucial for building a reliable and effective SMT-based GEC system. We also examined two methods which incorporate discriminative components into the generative SMT log-linear model. In the case of the second method — the first reported application of sparse features to GEC — our results significantly improve over the previous state-of-the-art in the field.

Streszczenie

Niniejsza praca doktorska dotyczy problemu automatycznej poprawy błędów językowych w tekstach pisanych przez osoby uczące się języka angielskiego jako języka obcego. Problem ten zbadano za pomocą metod tłumaczenia maszynowego. W celu zgromadzenia dodatkowych danych, opracowano metodę automatycznej ekstrakcji potencjalnych błędów językowych z historii edycji tekstu oraz stworzono największy publicznie dostępny korpus błędów. Zbadano automatyczne miary ewaluacji stosowane w dziedzinie pod kątem ich korelacji z ocenami ludzkimi poprzez wykonanie szczegółowego studium ewaluacji systemów do automatycznej korekty tekstu. Opracowany system, wykorzystujący metody statystycznego tłumaczenia maszynowego opartego na frazach, osiągnął najwyższe publikowane do tej pory wyniki na popularnym zestawie testowym CoNLL-2014 udostępnionym w ramach zadania organizowanego podczas Conference on Natural Language Processing w 2014 roku. W pracy pokazano jak istotne są właściwa optymalizacja modelu na podstawie przyjętej miary ewaluacji oraz zastosowanie nowych cech gęstych. Zbadano również dwie metody integracji algorytmów dyskryminacyjnych do generatywnego systemu tłumaczenia poprzez rozszerzenie modelu log-liniowego. Druga z metod — pierwsze zastosowanie cech rzadkich do zadania korekty tekstu — w sposób istotny rozszerza aktualny stan wiedzy w dziedzinie.

Acknowledgements

First and foremost, I would like to gratefully thank to my auxiliary supervisor Dr Marcin Junczys-Dowmunt for his help, support and motivation. He is a great mentor, teacher and colleague. I am also grateful to my supervisor Professor Krzysztof Jassem for his valuable advices, prominent patience and for giving me freedom to mature as a researcher. I thank to Dr Filip Graliński, who is responsible for introducing me to natural language processing. Finally, I am grateful to my girlfriend, my family and friends for their generous support.

Some parts of this thesis are based on the work partially founded by the Polish National Science Centre (Grant Np. 2014/15/N/ST6/02330).

Contents

Chapter 1

Introduction

This thesis considers the problem of automated Grammatical Error Correction (GEC) in texts written by non-native English speakers. The main goal of our research was to develop efficient algorithms and methods that can verify and improve grammatical correctness of text in a fully automatic manner.

1.1 Motivations

In recent years, automated grammatical error correction has grown in popularity as a part of the Natural Language Processing (NLP) field of research. One reason for this is a plurality of possible practical applications. Simple spelling and grammar checking components are built into numerous applications, such as text processors, email clients and web browsers, facilitating the creation of error-free texts. More sophisticated solutions are incorporated into systems for comprehensive proofreading and Computer-Assisted Language Learning (CALL). Moreover, fully automated text correction is used as a part of pre- or postprocessing in various NLP tasks, such as information retrieval, optical character recognition, automatic speech recognition, and Machine Translation (MT) (Bassil and Alwani, 2012, Habash, 2008, Perez-Cortes et al., 2000). Despite its popularity, the automated grammatical error correction is still far from being solved completely (Bryant and Ng, 2015).

In this thesis, we deal with errors made by English as a Second Language (ESL) learners. English is one of the most commonly-used languages; it is the third language in the world with the largest number of native speakers, and the first one according to the number of non-native speakers (Lewis et al., 2015). Also, most research on automatic error correction has studied errors produced by second language (L2) learners. There are significantly more tools and resources, such as part-of-speech taggers and parsers, developed for English than for any other language. Those allow NLP researchers to develop advanced algorithms and to produce comparable results.

One of the objectives of our research is to maintain the simplicity and clarity of created models and to make them language-independent to the extent possible. Recently, state-of-the-art GEC systems aimed to detect and correct the largest possible number of errors usually combine various algorithms and approaches developed separately for specific error types (Felice et al., 2014, Rozovskaya and Roth, 2014). This results in a high degree of complexity and makes the results more difficult to reproduce.

Our research is based on Statistical Machine Translation (SMT) approach. The modern formulation of the phrase-based SMT model allows incorporating new algorithms, or even other

approaches, at various levels of processing in a clear way. Such an approach, in which grammatical error correction is considered as a kind of machine translation, in this case from erroneous English to correct English, turned out to be severely underresearched, which motivated us to investigate further.

1.2 Goals and hypotheses

The main goal of this thesis is to develop efficient algorithms and methods for automatic correction of grammatical errors of various types produced by English learners. We hypothesize that the combination of generative phrase-based statistical machine translation models and discriminative classification components applied to automated grammatical error correction outperforms any of these separately used methods.

To verify this the following tasks have been defined:

1. Collecting examples of naturally-occurring grammatical and usage errors as training data for statistical data-driven approaches.
2. Choosing the most adequate automated evaluation metric that correlate best with human judgement.
3. Building an efficient automated grammatical error correction system based on phrase-based statistical machine translation framework.
4. Examine methods for integrating discriminative components into the phrase-based SMT model for automated grammatical error correction.

All these tasks are completed and described in the thesis.

1.3 Contributions of the thesis

This thesis gives a number of contributions to the field of automated grammatical error correction at both, theoretical and engineering levels.

Firstly, we provide a review of historical and state-of-the-art methods in the GEC field. Several approaches to automated error detection and correction are discussed and an exhaustive assessment of each approach is provided. This part of the thesis may also serve as a review of the field for a reader unfamiliar with GEC.

Secondly, we develop a method for the automatic extraction of potential errors from text edition histories. Using it, we create the WikEd Error Corpus — a publicly-available large corpus of corrective edits extracted from Wikipedia revisions.

Next, we present the first large-scale human evaluation of automated GEC systems. We use the produced system rankings to evaluate standard metrics for GEC and we show that the commonly used metric M^2 is well correlated with human judgements, and thus a reasonable choice as an evaluation metric.

Next, we reinvestigate the machine translation approach to automated grammatical error correction. New strong baselines for the field are created using SMT systems with proper optimization procedures and new task-specific dense features.

We examine two different methods that combine classifiers and SMT-based GEC systems. We incorporate discriminative components into the generative SMT model as a single feature function and by using task-specific sparse features.

Finally, the created WikEd Error Corpus[1], developed tools and data collected in the human evaluation study[2], and the SMT-based GEC systems[3] have been made publicly available and should be useful for other researchers.

1.4 Thesis outline

The structure of the thesis is as follows. In Chapter 2 we introduce automated grammatical error correction as a field of natural language processing. The scope of the research is defined and state-of-the-art approaches are described.

Next, in Chapter 3, we introduce applied data sets, including monolingual and error corpora. We also present a method for extracting naturally-occurring error examples from text revision histories.

Our choice of evaluation metrics and results of the first large-scale human evaluation of GEC systems are described in Chapter 4.

In Chapter 5, we describe a GEC system based on the phrase-based statistical machine translation models. An effective optimization procedure according to the chosen evaluation metrics and new task-specific features are presented.

Two different methods incorporating discriminative components into the generative phrase-based SMT model are studied in Chapter 6. We describe the integration of a cost-sensitive multi-class logistic discriminative classifier with label-dependent features, and sparse features derived from correction patterns.

We conclude in Chapter 7 by summarizing the key achievements and results, and formulating open questions for future work.

1.5 Publication notes

Some parts of this dissertation presents extended research described in the following publications:

- Grundkiewicz, R. (2013a). Automatic extraction of Polish language errors from text edition history. In *Text, Speech, and Dialogue — 16th International Conference, TSD 2013*, volume 8082 of *Lecture Notes in Computer Science*, pages 129–136, Plzen, Czech. Springer Berlin Heidelberg

- Grundkiewicz, R. (2013b). Errano: a tool for semi-automatic annotation of language errors. In *Proceedings of the 6th Language & Technology Conference*, pages 309–313, Poznan, Poland

- 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*

[1] https://github.com/snukky/wikiedits
[2] https://github.com/grammatical/evaluation
[3] https://github.com/grammatical/baselines-emnlp2016

Natural Language Learning: Shared Task, pages 25–33, Baltimore, Maryland. Association for Computational Linguistics

- 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 Przepiórkowski, A. and Ogrodniczuk, M., editors, *Advances in Natural Language Processing — Lecture Notes in Computer Science*, volume 8686, pages 478–490. Springer

- 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*, pages 461–470, Lisbon, Portugal. Association for Computational Linguistics

- Grundkiewicz, R. and Junczys-Dowmunt, M. (2015). Grammatical error correction with (almost) no linguistic knowledge. In *Proceedings of the 7th Language & Technology Conference*, pages 240–245, Poznan, Poland

- Junczys-Dowmunt, M. and Grundkiewicz, R. (2016). Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Austin, Texas. Association for Computational Linguistics

- Grundkiewicz, R. and Junczys-Dowmunt, M. (2017). Reinvestigating the classifiation approach to the article and preposition error correction. *Lecture Notes in Artificial Intelligence, Springer Verlag*, Human Language Technologies as a Challenge for Computer Science and Linguistics. To appear in the LTC 2015 post-conference volume

Chapter 2

Grammatical Error Correction

In this chapter we provide the scientific background for the rest of the thesis. Firstly, in Section 2.1, we introduce the notions of grammatical errors and automated error correction with focus on ESL errors. We also explain why GEC is a challenging task and where the difficulties come from. Section 2.2 describes in detail a number of approaches explored for the GEC field, and includes reviews of the most remarkable historical and state-of-the-art results presented in the literature. Then, in Section 2.3, we present the shared tasks dedicated to automated grammatical error correction, which have been organised in the past. We lay emphasis on the CoNLL-2014 shared task (Ng et al., 2014) as our experiments lie in the scope defined by this competition. The chapter is summarized in Section 2.4.

2.1 Automated grammatical error correction

The term *grammatical error* is ambiguous. It varies among languages as each language consists of unique grammar rules which, when violated, can cause grammatical errors. Non-native speakers are influenced by their first language (L1) and thus produce different errors than native speakers (Selinker and Gass, 1992). Some linguistic structures are misused depending on whether the writing style is formal or informal. The notion of grammatical error also varies among authors: linguists often form their definitions on linguistic source of errors (Ellis, 1994), while NLP researchers focus on a context or knowledge required for the proper detection and correction of the error (Kukich, 1992).

2.1.1 Grammatical errors

Grammatical errors are only one group of *language errors* (or linguistic errors). A general classification of language errors that fits to a variety of NLP applications has been proposed by Naber (2003) and consists of four categories:

- **Spelling errors** (or misspellings) result in words not belonging to the language. These are mainly mechanical errors that are also called *non-word errors* or *isolated errors* as no context information is needed to find them. Commonly-used spell checking techniques, such as finite state automata (Kukich, 1992) can detect such errors effectively.
- **Grammatical errors** violate grammar rules of the language. Grammar is defined as a set of systematic rules through which words and sentences are assembled to convey meaning

(Fraser and Hodson, 1978). This covers morphology, syntax and phonology of the language. These are mostly *real-word errors*, i.e. errors that consist in the wrong use of words that are a part of the language and meaningful, but in general they are not intended in the given context and violate grammar rules. Grammatical errors require an analysis of the surrounding context to be detected and corrected.

Context-sensitive spelling errors (Golding, 1995), i.e. spelling errors that originate from accidental usage of a word that belongs to the language, are also included in this group as they often appear as words with changed part-of-speech category.

- **Stylistic errors** (or usage errors) do not violate the syntax or morphology of a language, but conflict with the standard norm commonly accepted for a written discourse of specific type. Fraser and Hodson (1978) distinguish usage from grammar defining the former as alternative ways of writing acquired particular social status. Usage errors may be due to the use of uncommon words, overcomplicated structures, word repetitions or colloquial vocabulary.

- **Semantic errors** produce statements that are false according to general knowledge about the world. A wide context information, often in the form of extensive external knowledge, is required to correct such errors, which makes the task very difficult using the modern NLP methods. In fact, current solutions for automatic text correction rarely go beyond the scope of a sentence.

The notion of grammatical error used in NLP usually incorporates both grammatical and stylistic errors from the classification above. As pointed out by Leacock et al. (2010), a subset of spelling errors is often considered as grammatical errors, next to morphology, syntax and usage errors. In particular, these are spelling errors having source in the incomplete knowledge of morphology, such as incorrect word forms (e.g. *he writed* → *he wrote*), or contextual spelling errors (e.g. *form* → *from*).

2.1.2 Errors of English learners

As mentioned earlier, linguistic errors produced by native and non-native language users are typically different (Ellis, 1994). For example, a number of English learner errors, such as tense or modal confusions, or confusions of prepositions and articles, would be very rarely made by native English speakers. Some expressions and constructions applied by second language (L2) learners, even though acceptable for native speakers, are not likely to be used by them. Moreover, non-native speakers make errors more frequently. The majority of research in the field of grammatical error correction concerns errors made by non-native English speakers (Leacock et al., 2010).

Recently, NLP methods for ESL error correction are commonly evaluated on error-annotated corpora. A set of grammatical errors that is aimed to be detected and corrected is thus well-determined by annotations made in such corpora. As both corpora and tasks vary, this contributes to the diversity of language errors that are considered to be grammatical errors by the NLP community. In fact, if the goal is the correction of a variety of grammatical errors and a corpus-based evaluation is applied, NLP researchers do not have to reformulate the notion of the grammatical error as it is already defined by the task associated with the error corpus.

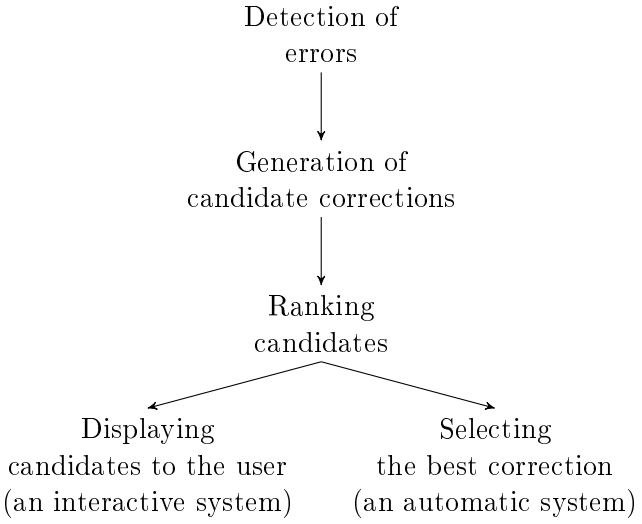


FIGURE 2.1: Automatic error correction process.

The set of grammatical errors that we attempt to correct in this work consists of all error types present in the NUCLE learner corpus (Section 3.1 in Chapter 3 provides a detailed description of this data set). Appendix A contains error categories annotated in the corpus and error examples.

2.1.3 Automatic error correction

Automatic error correction is the process that aims to find and correct linguistic errors in texts in an automatic manner. It is often split into error *detection* and error *correction*. Kukich (1992) distinguishes four phases of the task as presented in Figure 2.1.

Error detection is the process of finding the position and boundaries of an incorrectly used word or words in a text fragment. A undetected error cannot be corrected. If an error is detected, the next step is the generation of correction candidates. The third step is scoring and ranking of correction candidates starting from the most probable ones. The three phases of the process may be collapsed to a single part in modern systems, but they are usually distinguishable. In interactive systems the ranked list of candidate corrections is presented to the user. In a fully automated GEC systems, the error is replaced with the best correction candidate. GEC methods may vary in each phase. The change at the earlier phase has an impact on further stages and, finally, on the correction result.

Later in this work, we will use the term error correction to denote the whole correction process, including error detection. The distinction between detection and correction will be clearly stated when needed.

2.1.4 Difficulties in error correction

There are two sources of difficulties for automatic grammatical error correction: the hard-to-code linguistic nature of grammatical errors and a high level of technical complexity.

Natural language is ambiguous, which has an impact on the nature of linguistic errors. The most significant aspects that make the GEC task challenging are the following:

- Multiple corrections for a single error are possible. Many errors can be corrected in several ways, for example:

Above all, life is more important than [*secret* → *secrets/secrecy/a secret*].

Alternative corrections are possible due to the lack of context required to decide which correction is the most accurate, or due to the ambiguity of a language. For instance, the choice of many articles or determiners depends on personal preferences of a proofreader. Various corrections may require changes in different parts of a sentence, e.g. a subject-verb agreement error can be corrected by modifying either the subject or the verb.

Ambiguous corrections are a handicap for automatic evaluation methods, which usually compare the system output against the human annotator's choice (evaluation difficulties are discussed in Section 4.1).

- Grammatical errors occur with a low frequency. Error frequency is usually measured compared to the total number of sentences, total number of words, or the number of words that are specific to a given error type.

  Many sentences do not include any spelling, grammatical or stylistic errors, and the vast majority of words in a text are used correctly. The number of sentences that contain one or more errors varies between 35% and 85% depending on ESL corpus and dataset (Bryant and Ng, 2015, Dahlmeier et al., 2013, Yannakoudakis et al., 2011). The number of incorrectly used words is ca. 6–15%. Due to the low error rates a GEC system has to be very cautious, so that it would not introduce new errors into correct text fragments. Moreover, some error types occur with low frequency in text data, which make them difficult to learn for data-hungry statistical approaches.

- Error types and error distribution highly vary among writers and data sets. Error corpora are created from texts written by ESL learners with various levels of language proficiency and various native languages (Rozovskaya and Roth, 2010a). Moreover, as some corrections might be optional or performed in different ways, the annotations and their numbers vary among annotators for the same data set (Bryant and Ng, 2015). This is especially true for stylistic errors. For example, only two of ten annotators flagged the word *execute* as incorrect in the following sentence[1]:

  A good law should be clear and easy to [*execute* → *enforce*].

  The annotation guidelines can also influence the annotator's choices (Sakaguchi et al., 2016).

  These ambiguity issues pose challenges not only for evaluation, but also for the development of GEC systems. Statistical approaches, such as classification or statistical machine translation, are highly dependent on the distribution in the training data. Systems are often tuned on development sets with predefined error frequency and distribution. If, for instance, the error frequency in the test set is significantly lower than in the tuning set, the GEC system will be prone to produce many false positives, and vice versa, if error frequency in the test set is much higher than in the tuning set, many errors may be missed by such a system (Cahill et al., 2013, Rozovskaya and Roth, 2010c).

- Some errors depend on distinct word choices. An existence of a particular word may influence the choice of a another long-distance word. A particular word in the sentence

[1] The example comes from the CoNLL-2014 Test Set annotated by ten annotators. More detailed description of the dataset is presented in Chapter 3, Section 3.1.

might dictate the word choice much later, especially in complex sentences that consist of multiple clauses. The detection of the error might require the analysis of a wide context. Subject-verb agreement errors are the example:

The scent of red sweet apples and cinnamon sticks [*are* → *is*] *present in the wine.*

The knowledge of sentence syntax and/or dependency relationships between words from a syntax or dependency parser might be useful to detect such an error. The context that has to be taken into consideration is even broader when the error is related to word choices in preceding sentences. A number of methods, e.g. statistical machine translation, limit the processing to a single sentence, which means that the detection and correction of inter-sentence errors is unlikely by design.

- More than one error of any type might occur in a sentence. Multiple errors in a sentence may lead to the occurrence of an error in a context used in the detection of another error. This problem is solved by correcting contexts on the fly in the GEC processing or training on the context including erroneous constructions instead. This is usually not the case if systems are trained on native texts that are mostly error-free.

  This phenomenon also indicates that in the case when different components are used to detect different error types separately, a method combining these corrections is advisable.

- One erroneous word may manifest multiple errors. For instance, the incorrect form of the word may be due to both spelling and subject-verb agreement error, e.g.:

  An example is water, which is a good renewable [*resuorces* → *resource*] *and is plentiful.*

  The successful correction of the spelling error may result in a word that still contains the grammatical error (e.g. *resuorces* → *resources*).

Automated grammatical error correction also poses a number of challenges as a natural language processing task due to the high technical complexity. Many high-performance systems use components that execute many NLP subtasks, such as sentence segmentation, word tokenization, POS tagging, chunking and shallow parsing, named entity recognition, or spelling error correction. That complexity might be a source of the following issues:

- Most NLP tools assume error-free input texts. As many of these tools are developed on correct texts, their performance is often suboptimal on texts with language errors.

- The lower the efficiency of each system's component, the lower the final performance. Even tools executing well-studied tasks, such as part-of-speech tagging, do not achieve perfect results. Moreover, the limitations of individual components stack over pipeline processing. For example, incorrectly assigned POS tags have a negative impact on the performance of shallow parsing, etc.

- Statistical NLP tools can hide some errors or make them more difficult to detect. For instance, if a statistical POS tagger has been trained only on error-free data, it might assign an incorrect part-of-speech tag to words that cause real-word errors.

- NLP tools are complex on their own and this causes additional difficulties in their integration. For instance, tools developed in different NLP centres often use different tokenization schemas or tag sets.

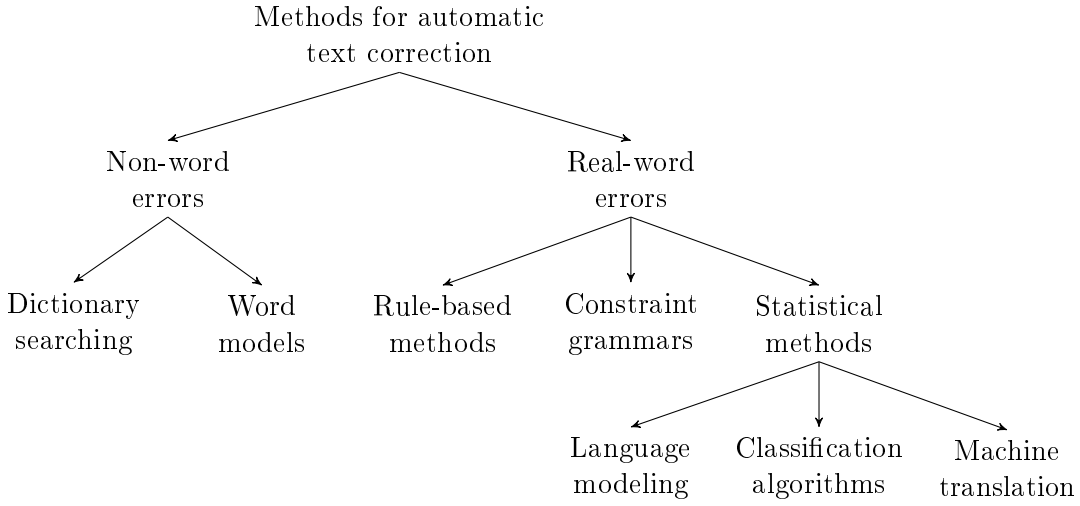


FIGURE 2.2: Classification of text correction approaches based on error types.

More challenges in grammatical error correction concern the evaluation of GEC systems. Such issues are discussed in Section 4.1.

2.2 Approaches to automated grammatical error correction

In last decades, a number of approaches have been proposed for automated grammatical error correction. The general classification of the most successful developed methods is presented in Figure 2.2. It distinguishes methods for non-word and real-word errors (Section 2.1.1). Standard techniques for detection of isolated errors are based on dictionary lookup or analysis of character n-grams (Kukich, 1992). As non-word errors are not targeted in this thesis, these techniques are not discussed in the subsequent parts of the chapter.

Real-word errors can be corrected using rules that take into account the context around an error. Methods that are based mostly on manually created rules are called *rule-based* methods. *Grammar-based* methods are built on top of formal grammars and use modified parsing algorithms to detect and correct grammatical errors Leacock et al. (2010). Rules can be also acquired transparently from statistical data-driven methods. The most commonly used machine learning approaches to GEC are *language modeling*, *classification algorithms* and *statistical machine translation*. It should be noted that some non-word errors can also be effectively detected and corrected by statistical methods commonly used for real-word errors if they occur frequently enough in training data.

In the rest of this section, we describe selected approaches, including the most impactful historical methods and the state-of-the-art solutions, which mostly follow statistical and/or hybrid approaches.

2.2.1 Rule-based methods

The first grammar checking tools, such as the Unix Writer's Workbench (MacDonald et al., 1982) or EPISTLE and CRITIQUE (Heidorn et al., 1982), used hand-crafted rules and pattern-matching techniques. The most widely used grammar checker nowadays from Microsoft Word

```
<rule name="Agreement error: past participle without 'have'">
  <pattern>
    <token regexp="yes">([wc]|sh)ould|might|must</token><marker>
    <token postag="VBN"><exception postag="VB[PD]" postag_regexp="yes"/>
      </token></marker>
  </pattern>
  <message>
    Did you mean <suggestion>have <match no="2"/></suggestion> or
    <suggestion><match no="2" postag="VB"/></suggestion>?
  </message>
  <short>Possible grammatical error</short>
  <example correction="have been|be">
    You could <marker>been</marker> from Russia.
  </example>
  <example>They should break away from Russia.</example>
</rule>
```

FIGURE 2.3: Example of a Language Tool error-matching rule.

text editor (Heidorn, 2000) relies mostly on a rule-based approach. Another recent example is LanguageTool[2] (Miłkowski, 2010), which has been initially developed by Naber (2003).

Modern rule-based GEC systems commonly support such features as:

- Pattern matching with regular expressions.
- Alternative correction suggestions.
- Analysis of the input text at several linguistic levels providing results of morphosyntactic analysis, noun-phrase chunking (Miłkowski, 2010) or dependency parsing (Mozgovoy, 2011).
- Basic logic and set operations on patterns and rules, e.g. the negation, union or intersection of rules.
- Generation of suggestions with synthesizer of inflected forms.

An example of the agreement error detection rule from LanguageTool is presented in Figure 2.3. The rule matches a past participle (e.g. *been*) not preceded by the auxiliary verb *have* but preceded by other auxiliary verbs, such as *would* or *should*, and suggests two alternative corrections (e.g. *have been* and *be*).

Although error-matching rules are usually manually created from frequently observed patterns of errors in text corpora (Andersen et al., 2013, Miłkowski, 2010), these can be also acquired by automatic or semi-automatic methods. For example, Miłkowski (2009) proposed to use a transformation-based learning algorithm to automatically acquire symbolic rules for LanguageTool.

Since the 90s, in line with general trends in NLP, the rule-based approach has been replaced by statistical and data-driven methods due to the increasing availability of annotated corpora. Despite this, rule-based components may still be present in GEC systems using combined approaches. For example, Felice et al. (2014) apply a rule-based module suggesting specific

[2] https://www.languagetool.org/

corrections for frequent errors from the Self-Assessment and Tutoring (SAT) system (Andersen et al., 2013) as the first step in their hybrid system. Wu et al. (2014) and Lee and Lee (2014) attempt to correct subject-verb agreement errors using rule-based methods and apply different approaches for other errors.

The main advantages of the rule-based approach are (Naber, 2003):

- Errors for which context is clearly defined are easy to detect and correct regardless of their frequency in the error corpora.
- Rules can be created incrementally. A rule-based grammar checker works immediately after the first rule is implemented.
- Easy configuration of the system: each rule has its own description and error message, and moreover, each rule can be turn on or turn off individually.

On the other hand, the disadvantages of this approach include:

- Development of hand-crafted rules is costly and time-consuming, and usually needs to be carried out by linguist specialists.
- High language-dependency as rules need to be developed for each language separately.
- Difficulty of tackling open-class errors, for example inflectional errors in highly inflectional languages.

2.2.2 Formal grammars

In the past, researchers attempted to build rules on top of computational grammars in so-called *grammar-based approach*. A grammar needs to be error-tolerant to parse a sentence with grammatical errors. Several techniques have been proposed to meet this requirement (Leacock et al., 2010, Chapter 2).

For example, Douglas and Dale (1992) introduce relaxations into the grammar by making certain agreement constraints optional. Dini and Malnati (1993) over-generate parse trees that violate grammatical constraints and rank them in order from the highest number of satisfied constraints. Holan et al. (1997) locate the source of the error using an algorithm that compares available parse trees with inconsistencies. The grammar checker of Park et al. (1997) is based on error-rules that are applied if all standard grammar rules fail. A similar approach is based on rules that allow to parse specific erroneous constructions (so-called mal-rules) in the input text (Bender et al., 2004, Schneider and McCoy, 1998).

As noted by Crysmann et al. (2008), since coverage of developed grammars is never exhaustive, GEC systems based on a formal grammar also have difficulty to distinguish between non-covered grammatical and truly ungrammatical sentences. Conversely, since grammars often overgenerate, a successful parse does not guarantee well-formedness either.

2.2.3 Language modeling

A statistical (or probabilistic) *language model* (LM) is a probability distribution over a sequence of characters or words (Manning et al., 2008). LM assigns probability $P(t_1, \ldots, t_m)$ to the sequence of words $t_1, \ldots, t_m$ of length m, which attempts to reflect how frequently and how

probably the given sequence occurs in the language. Probabilities are estimated on a large corpus, which ideally consists of error-free texts.

The most popular statistical language models are n-gram language models (Shannon, 1951). An *n-gram* is a sequence of tokens $t_1, \ldots, t_n$ of length n. If n is equal to 1, 2 or 3, an n-gram is called unigram, bigram or trigram, respectively. An n-gram language model estimates the probability p of a sequence of tokens T based on the previous $n-1$ tokens using n-th order Markov property (Markov, 1960):

$$p(T) \approx \prod_{i=1}^{|T|} p(t_i|t_i, \ldots, t_{i-n+1}).$$

In a naive approach, an error is detected if a very improbable word within a sentence (according to the language model) is found. The correction is made by a substitution of the flagged error with a word that increases the overall probability of the sentence. Usually the correction candidate is in a short edit distance to the erroneous word. Language models allow to detect both non-word and real-word errors.

One of the first attempts to use language models for GEC is described by Atwell (Atwell, 1987, Atwell and Elliot, 1987). The author used a part-of-speech tag language model to flag unlikely POS tag transitions in the input text and proposed several methods to determine when an error should be diagnosed. He experimented with using manually-created error-likelihoods which measure how frequently each tag pair occurs in an error, finding an optimal threshold for low absolute likelihoods for POS tags, or adding error-tags to lexical entries that, when used, indicate the occurrence of an error. Chodorow and Leacock (2000) use mutual information and Chi-square statistics to determine which sequences of POS tags and function words are unusually rare with regard to a large well-formed corpus, and therefore are likely to be ungrammatical in English. A language model based on a generative statistical parser is used by Turner and Charniak (2007) to determine the most likely grammatical determiner to be used in a noun phrase. Stehouwer and van Zaanen (2009) investigate various language models to determine which word, from a predefined set of confusable words[3], is most probable in a given context. The proposed systems use a uniform or weighted linear combination of different n-grams around the confusable word and different back-off strategies to counteract the missing word sequences in the trigram language model. More recent research of Hdez and Calvo (2014) focuses on examining a language model based on syntactic trigrams and bigrams extracted from dependency trees generated from English Wikipedia.

The effectiveness of a probabilistic language model highly depends on size and quality of text corpora used to estimate probabilities (Liu and Curran, 2006). A number of researchers (Elghafari et al., 2010, Gamon and Leacock, 2010, Gamon et al., 2009, Hermet et al., 2008, Tetreault and Chodorow, 2009, Yi et al., 2008) decided to use n-gram counts extracted from search engines as a language model approximation. Commonly used sources of n-gram data sets are Microsoft Web N-gram Services (Wang et al., 2010) and Google Web 1T N-gram Corpus (Brants and Franz, 2006).

Language models are commonly used in GEC systems combined with other methods. In recent hybrid systems, the main purpose of a language model is to rank or validate correction decisions (Felice et al., 2014, Lee and Lee, 2014, Wu et al., 2014). LMs are also integral parts of GEC systems based on statistical machine translation approach.

[3]The notion of a confusion set will be defined in the next section.

The most significant advantages of the LM-based approach for GEC are as follows:

- Language models need only plain text resources that are widely available and plentiful today.
- Language models can be easily combined with other methods.

On the other hand, the disadvantages of this approach include:

- N-grams that do not occur in training data and thus have zero probability have to be handled with smoothing methods.
- The distinction between rare n-grams and ungrammatical n-grams has to be modeled.

2.2.4 Classification

The basis for the *classification* approach are supervised machine learning methods. Error correction is seen as a lexical disambiguation task or word selection task (Golding, 1995, Roth, 1998). The ambiguity between correction candidates is modeled by pre-defined *confusion sets* (or *candidate sets*) containing commonly confused words.

	about *at* *by* *for* *from* *in*		*about* *at* *by* *for* *from*	
A study	*of*	New York University	*in*	2010 shown that patients (...)
	on *to* *with*		*of* *on* *to* *with*	

FIGURE 2.4: Example of classification approach for preposition correction. The confusion set consists of ten most frequent prepositions: {*about, at, by, for, from, in, of, on, to, with*}. Source words are underlined.

Figure 2.4 illustrates the classification task given a confusion set that consists of ten most frequent prepositions {*about, at, by, for, from, in, of, on, to, with*}. A word that belongs to the confusion set and that has been encountered in the input text is called a *source word* (prepositions *of* and *in* in the example). The task of the pre-trained *classifier* is to decide for each source word, which of the finite number of possible alternatives from the candidate set is the most accurate in the given context. The context is represented by neighbouring words from the sentence in which the source word appears, and modeled by *context features*. The classification algorithm is trained on a set of labeled examples extracted from an error-annotated corpus.

Applications of classifiers to GEC vary in three aspects: the selection of confusion sets, the design of context features, and the choice of classification algorithms.

Confusion sets The classification approach has been initially applied to context-sensitive spelling error (CSS) correction (Golding, 1995, Golding and Roth, 1996, 1999, Golding and Schabes, 1996). In this task, confusion sets commonly consist of a small number of words that are homophones (e.g. {*whether, weather*}), have similar spelling (e.g. {*than, then*}), or share some grammatical functions (e.g. {*between, among*}). A substantial amount of research concerns article and preposition errors produced by ESL learners (Dahlmeier and Ng, 2011, De Felice and Pulman, 2008, Gamon, 2010, Gamon et al., 2008, Han et al., 2006, Izumi et al., 2003, Rozovskaya and Roth, 2014, Tetreault et al., 2010, Tetreault and Chodorow, 2008). The HOO 2012 shared task (Dale et al., 2012) was dedicated to these two types of errors. They constituted two of five error types being a subject of the CoNLL-2013 shared task Ng et al. (2013) (more detailed description of shared tasks is included in Section 2.3).

Preposition and article errors are examples of closed-class errors, which can be effectively modeled by finite confusion sets. However, the classification-based approach has also been used for open-class errors, such as subject-verb agreement or noun number errors (Jia et al., 2013, Rozovskaya et al., 2014a,b, van den Bosch and Berck, 2013). Confusion sets are generated on the fly for each word with a specific part-of-speech tag based on linguistic properties of particular errors.

Context features The set of context features is usually developed specifically for each error type. Features used for CSS correction are usually based on n-gram and bag-of-word features. More sophisticated, linguistically-motivated features are commonly designed for articles or determiner errors, and for open-class errors. However, for some error types, such as preposition errors, good results are achieved with feature sets of n-gram features only (Rozovskaya and Roth, 2014).

The majority of feature sets that have been described in the literature consist of the following features:

- Context-word features, which test for the presence of a particular word within a fixed-size window around the source word (Golding, 1995, Golding and Roth, 1999).
- N-grams of tokens or tags (e.g. part-of-speech tags), which include or are adjacent to the target word (Rozovskaya and Roth, 2010b).
- Syntactic features requiring at least a noun-phrase or verb-phrase chunking (Chodorow et al., 2007, De Felice and Pulman, 2008, Tetreault et al., 2010).
- Features that mix various tags in a single n-gram (Fossati and Di Eugenio, 2007, Tetreault and Chodorow, 2008).
- Linguistically-motivated hand-crafted features, which combine words and POS tags based on the information from a chunker (De Felice and Pulman, 2008, Rozovskaya et al., 2013, 2014a, 2012).
- Morphology features, such as the number or the countability of a head noun (De Felice and Pulman, 2008, Lee, 2004).
- Features including semantic information, for example hypernyms or WordNet categories (De Felice and Pulman, 2008, Hirst and Budanitsky, 2005, Lee, 2004).

Rozovskaya and Roth (2010c) introduced the *selection* and *correction training paradigms*. The difference between the two is whether the source word is taken into consideration during training or not. In the selection paradigm, the source word is not used as a feature, so models for ESL error correction are usually trained on native English data, and then applied to non-native texts. The correction paradigm, in turn, makes the use of original author choice's and thus requires error-annotated data. Recently, as large annotated ESL corpora have become available, the correction paradigm is used more commonly (Cahill et al., 2013, Grundkiewicz and Junczys-Dowmunt, 2015, Rozovskaya and Roth, 2010c, 2014), and gives better results than training with the selection paradigm, which used to be more popular beforehand (Chodorow et al., 2007, De Felice and Pulman, 2007, 2008, Gamon, 2010, Han et al., 2006, Tetreault and Chodorow, 2008).

Classification algorithms So far, a substantial number of classification algorithms have been used in GEC. Examples are: decision trees, Naive Bayes classifier, Winnow algorithm, averaged perceptron, linear and logistic regression, maximum entropy classifier, or support vector machines. The choice of the classifier, in contrast to the choice of a feature set, is generally independent of the confusion set and the type of targeted errors.

Classification-based methods have been shown to produce the state-of-the-art results for the correction of a number of error types. These are among the most actively studied approaches to GEC in the last decade as the broad range of improvements to classification models are possible. The most significant advantages of this approach are as follows:

- A wide range of well-known classification algorithms and machine learning techniques can be applied.
- It is possible to target specific error types by designing various confusion sets.
- When training data is available, the method is less costly and more general than the rule-based approach.
- The method is mostly language-independent, except for the need of annotated data for a given language.

Despite the popularity of classification methods, building a separate model for each grammatical error is a complex task (Leacock et al., 2010, Chapter 2). The main limitations include:

- Only specific closed-class errors can be directly modeled by finite confusion sets. Modelling open-class errors is more challenging.
- A large amount of annotated text data is needed for training purposes. Size and quality of the data set highly influence the performance.
- A separate model usually needs to be developed for each confusion set and error type, which might include complex feature engineering.
- The problem of word insertions has to be handled separately, beyond the classification model.

2.2.5 Statistical machine translation

Automated grammatical error correction can also be considered as a kind of machine translation task. The translation is performed from text with errors — interpreted as the source language — into error-free text, which is treated as the target language.

In phrase-based statistical machine translation, for the input sentence S, the suggested correct sentence $\hat{T}$ maximizes the conditional probability over possible corrections (Koehn, 2010). The probability is calculated using a log-linear model as the weighted combination of feature functions $h_i(T|S)$:

$$\begin{aligned}\hat{T} &= \arg\max_{T} p(T|S) \\ &\approx \arg\max_{T} \exp\big(\sum_{i=1} \lambda_i \log h_i(T|S)\big).\end{aligned}$$

Typical feature functions are the translation model learnt from a sentence-aligned parallel corpus and the language model estimated on error-free texts. Weights λ_i should be learnt accordingly to the evaluation metric (Och and Ney, 2002)[4].

For the first time this approach has been applied to grammatical error correction by Brockett et al. (2006). The authors use SMT to correct countability errors for a set of 14 mass nouns that pose problems to Chinese ESL learners. For this very restricted task they achieve the results of 61.81% corrected mistakes and show that their system can beat the Microsoft Word 2003 grammar checker. A GEC system based on the Moses SMT toolkit[5] (Koehn et al., 2007) that corrects grammatical errors of learners of Japanese is described by Mizumoto et al. (2011). This work is continued for English in Mizumoto et al. (2012) and the effect of learner corpus size on various types of grammatical errors is investigated. Dahlmeier and Ng (2012a) introduce a custom beam-search decoder that incorporates discriminative classifiers for specific error categories such as articles and prepositions.

The most notable SMT-based systems that participated in the CoNLL-2013 shared task are Yuan and Felice (2013) and Yoshimoto et al. (2013) (a detailed description of the CoNLL shared tasks dedicated to GEC is presented in Section 2.3). Yuan and Felice (2013) apply POS-factored SMT model to five error types and experiment with training data containing generated artificial errors. Improvements over the baseline are small, but their approach to generate errors looks promising. Yoshimoto et al. (2013) use Moses for prepositions and determiners, but for other error types they find that classifier-based approaches and treelet language models perform better. The winning system (Felice et al., 2014) in the CoNLL-2014 shared task uses a hybrid approach that pipelines a rule-based and an SMT error correction system augmented by a large web-based language model. The AMU system (Junczys-Dowmunt and Grundkiewicz, 2014), placed third, is our contribution that is based on a phrase-based SMT system. Wang et al. (2014) trained factored models incorporating word stem, prefix, suffix and part-of-speech information.

More recent works which also rely on SMT systems exploit n-best list re-ranking methods (Hoang et al., 2016, Mizumoto and Matsumoto, 2016, Yuan et al., 2016) or new feature functions, such as a feed-forward neural translation joint models (Chollampatt et al., 2016a). However, most of the improvement over the CoNLL-2014 shared task of these works is due to using

[4] Optimization of model weights is described in Chapter 5.

[5] http://www.statmt.org/moses/

the parameter tuning tools that we introduced in Junczys-Dowmunt and Grundkiewicz (2014)[6]. Besides, Yuan and Briscoe (2016) show that recent models of Neural Machine Translation (NMT) (Bahdanau et al., 2014) can be effective for the GEC task.

There are several advantages of using the statistical machine translation comparing to other approaches to GEC:

- The lack of restriction to specific error types causes that a broad range of errors can be corrected, including open-class errors.
- Phrase-based SMT system can naturally correct errors within phrases, not only for individual words.
- SMT-based system can simultaneously correct interacting errors within a sentence.
- The parallel data with the corrected versions of sentences is sufficient for training purposes, i.e. no annotations of error types are required.
- Monolingual data, which is more easily available than parallel data, can be easily incorporated into SMT systems in the form of language models.
- The machine translation approach is relatively easy to adapt to other languages.
- Building an SMT system does not require expert linguistic knowledge on grammatical error correction.

Disadvantages of machine translation approach include:

- The performance of the SMT system highly depends on the amount and quality of training data.
- Errors that have not been observed in training data cannot be corrected. This affects, for example, spelling errors.
- Correction of errors within long-range dependencies in a sentence may be difficult for phrase-based SMT systems.
- Controlling and manipulating the correction of individual error types is difficult when training was not restricted to them.

2.2.6 Combined approaches

Various methods seem to be better suited for the correction of specific error types. GEC systems that use separate components to tackle different error types have to incorporate techniques that combine the outputs from these components. Common approaches include the pipelining of system components, where one error type is corrected after the other sequentially, re-scoring of correction candidates from one component by the other, or combining individual results.

In Gamon et al. (2008, 2009), language models provide additional information to filter out spurious suggestions from error-specific classifiers. Language models and classifiers are combined in a meta-classification approach in Gamon (2010). Ehsan and Faili (2013) combine an SMT system trained on artificial erroneous sentences with a rule-based grammar checker in an interactive

[6] Parameter tuning according to the GEC-specific evaluation metric is described in Chapter 5.

system for English and Farsi by aggregating correction candidates from both components. The authors show that the two approaches are complementary, and the best performance is achieved by the hybrid system. Boroş et al. (2014) use rule-based methods to detect errors, but the correction is based on LM scores. Felice et al. (2014) explore various strategies for the combination of rule-based methods with an SMT system and language models. Their best hybrid system, which placed first in the CoNLL-2014 shared task, pipelines a rule-based module with phrase-based SMT system producing candidate corrections, which are then ranked by a web-based language model and filtered by error type. After the shared task, Susanto et al. (2014) published work on GEC systems combinations. They combine the output from a classification-based system and an SMT-based system using Multi-Engine Machine Translation (MEMT) (Heafield and Lavie, 2010) — an SMT-specific tool for the combination of multiple MT outputs. The method combines single-best outputs from multiple independent systems to form an n-best list of combined translations that improve over individual systems. In the very recent work, Rozovskaya and Roth (2016) show the complementarity of the classification and SMT approaches by pipelining two systems. New state-of-the-art results on the CoNLL-2014 test set are achieved if a spell checker and classifiers are applied before the SMT system to the input text.

2.3 Shared tasks on GEC

In recent years, many of the advances in GEC emerged from various shared tasks. These are open competitions, where participating teams are encouraged to develop systems for grammatical error correction within constraints defined by the task organizers. They provide data sets and evaluation frameworks, which allow for an objective and comparable evaluation of the developed methods.

Shared tasks differ from each other by focusing on different error types or targeting different domains and languages.

2.3.1 The HOO shared tasks

The competitions on automatic grammatical error correction was started by the Helping Our Own (HOO) (Dale and Kilgarriff, 2010, 2011) pilot shared task organised in 2011. The aim of the task was to develop automated methods that assist authors within the NLP community with the writing of scientific papers. Annotated text fragments from papers that had previously been published in the proceedings of a conference or a workshop of the Association for Computational Linguistics (ACL) were used as training and test data. The performance of the six participating systems was evaluated using an F-score measure (Rijsbergen, 1979) on detection, recognition and correction levels.

The second edition of the task — the HOO 2012 (Dale et al., 2012) — was focused on detection and correction of determiner and preposition errors made by non-native speakers of English. The organizers adapted the publicly-available CLC FCE Dataset (Yannakoudakis et al., 2011) as training and test data for the shared task. 14 teams participated in the task. On a blind test set, the UI system (Rozovskaya et al., 2012) achieved the highest F-score for detection and recognition subtasks, the NU system (Dahlmeier et al., 2012) won the correction subtask. Both systems used pipelined classifiers to tackle both error types.

2.3.2 The CoNLL shared tasks

The most impactful competitions for ESL grammatical error correction were the two CoNLL shared tasks (Ng et al., 2014, 2013) organised as a part of the Conference on Natural Language Learning (CoNLL) in 2013 and 2014.

The goal of the shared tasks was to evaluate algorithms and systems for automatic correction of grammatical errors in English essays written by L2 learners of English. A participating system had to detect the grammatical errors that occur in the input texts, and return the corrected texts in a fully automatic manner. The CoNLL-2013 shared task was devoted to the correction of errors within the five selected error categories (article or determiner, preposition, noun number, verb form and subject-verb agreement errors), which account for from one-third to one-half of all errors in the provided data sets. The 2014 edition targeted grammatical errors of all types occurring in the essays, grouped into 28 categories. The lack of restriction for specific error types introduced a more natural scenario for the GEC task, which has been followed in further competitions, such as QALB (Mohit et al., 2014a, Rozovskaya et al., 2015) or AESW (Daudaravicius, 2015).

Participating teams were given training data with manually annotated corrections of grammatical errors and were allowed to use additional publicly available data for development. The common training data set made available by the organisers was the National University of Singapore Learner Corpus (NUCLE) (Dahlmeier et al., 2013) (see Section 3.1 for a more detailed description of the corpus). The corrected system outputs were evaluated using the MaxMatch (M^2) scorer (Dahlmeier and Ng, 2012b) over blind test data annotated similarly to provided training data. Between the CoNLL-2013 and CoNLL-2014 shared tasks, the organizers adjusted the F_β-score from $\beta = 1.0$ to $\beta = 0.5$, weighting precision twice as much as recall (see Section 4.2 for the description of M^2 metric).

During the CoNLL-2013, most of the participating systems used the machine learning-based classifier approach, including the winning system UIUC (Rozovskaya et al., 2013), some used rule-based or language modeling components. Three systems STEL (Buys and van der Merwe, 2013), CAMB (Yuan and Felice, 2013) and TOR (Wilcox-O'Hearn, 2013) (ranked as six, eight and eleven, respectively) used machine translation to correct all five error categories, whereas the fourth system NARA (Yoshimoto et al., 2013) used this approach for preposition and determiner errors only.

Thirteen system submissions took part in the CoNLL-2014 shared task. The official results are presented in Table 2.1, where systems that used a partially (*) or fully (**) SMT-based approach are asterisked. Among the top-three positioned systems, two submissions — CAMB (Felice et al., 2014) and AMU (Junczys-Dowmunt and Grundkiewicz, 2014) — were partially or fully based on SMT. The second system, CUUI (Rozovskaya et al., 2014a), was a classifier-based approach, which used various classifiers to correct specific error types and did not attempt to tackle the whole range of errors. The best system (CAMB) used a hybrid approach that pipelines a rule-based and an SMT error correction system augmented by a large web-based language model. The AMU system, ranked in the third place, is our contribution based on a phrase-based SMT system.

In this thesis, we generally follow the constraints established during the CoNLL-2014 shared task, which allows for a competitive comparison, not only with systems participating in the shared task, but also with a number of successive works that have evaluated their methods on the official CoNLL-2014 test set.

Rank	Team ID	P	R	$M^2_{0.5}$
1	CAMB*	39.71	30.10	37.33
2	CUUI	41.78	24.88	36.79
3	AMU**	41.62	21.40	35.01
4	POST	34.51	21.73	30.88
5	NTHU*	35.08	18.85	29.92
6	RAC	33.14	14.99	26.68
7	UMC**	31.27	14.46	25.37
8	PKU	32.21	13.65	25.32
9	NARA	21.57	29.38	22.78
10	SJTU	30.11	5.10	15.19
11	UFC	70.00	1.72	7.84
12	IPN	11.28	2.85	7.09
13	IITB*	30.77	1.39	5.90

TABLE 2.1: The official CoNLL-2014 shared task results.

2.3.3 Other competitions

Similar competitions focused on grammatical error correction have been organised for languages other than English. In 2014 and 2015, two QALB shared tasks on automatic text correction for Arabic (Mohit et al., 2014a, Rozovskaya et al., 2015) were organized in a similar manner to the CoNLL shared tasks. Whereas QALB-2014 addressed errors produced by native Arabic speakers in online comments, the next year's edition offered an additional track focused on errors found in essays written by language learners. Participating teams were provided with training and development data and were free to make use of additional publicly available resources. The results were computed over blind data sets using the M^2 scorer and $F_{1.0}$ measure.

Another competition for non-English language is a shared task on Chinese Grammatical Error Diagnosis (CGED) (Lee et al., 2015, Yu et al., 2014) organised during the Workshop on Natural Language Processing Techniques for Educational Application (NLP-TEA) since 2014.

There were also tasks not directly focused on the grammatical error correction, but related to the field and influence it to some extent. The Automated Evaluation of Scientific Writing (AESW) (Daudaravicius, 2015) shared task and the annual shared task on Automatic Post-editing (APE) (Bojar et al., 2016) are the most substantial examples.

2.4 Summary

This chapter introduced the field of automated grammatical error correction. Recent formulations of the GEC task put no restriction on particular error types. The objective is the detection and correction of a whole range of grammatical errors that can be produced by language users. We reviewed several approaches proposed for GEC and pointed out their strengths and weaknesses.

Recently, data-driven methods lead the way in GEC recently. The latest studies and presented advantages of statistical machine translation approach seem to justify the choice of SMT models as the basis for our research. In our experiments, we will follow the settings introduced during the CoNLL-2014 shared task on the correction of grammatical errors made by non-native learners of English.

Chapter 3

Data Sets

Although some types of errors, for instance subject-verb mistakes, can be corrected using heuristic rules, others, like article or preposition errors, are difficult to correct without substantial amounts of corpus-based information (Leacock et al., 2010). This is especially true for data-driven approaches, such as supervised classification (Cahill et al., 2013) and statistical machine translation (Mizumoto et al., 2012). Error-annotated corpora are also essential for the correction training paradigm (Rozovskaya and Roth, 2010c).

In this chapter, we discuss the types and sources of data used to build GEC systems. We review selected error corpora and monolingual data in Section 3.1. Section 3.2 describes a language-independent method of edition mining from Wikipedia revision histories, which lead to the building of the WikEd Error Corpus (Grundkiewicz and Junczys-Dowmunt, 2014)[1] — the largest publicly available error annotated data set with possible applications to sentence paraphrasing, spelling correction, and grammatical error correction.

3.1 Error corpora and monolingual data

The leading approaches to gathering error corpora presented in the literature are: manual annotation of learners' writings, artificial errors generation within well-formed sentences, and the extraction of errors and their corrections from text edit histories. Another possibility are social networks for language learners.

Besides parallel data sets, monolingual data can also be useful for GEC.

3.1.1 Learner's corpora

Compared to multilingual translation corpora which today are plentiful or can be easily collected[2], genuine error corpora are not easy to come by. As noted by Leacock et al. (2010), even if large quantities of students' writings are produced and corrected every day, only a small number of them is archived in electronic form.

Most of the available error-annotated corpora has been created from the learners' writings. The most popular publicly-available data set today, serving as a standard resource for empirical approaches to grammatical error correction, is **the NUS Corpus of Learner English** (NUCLE) (Dahlmeier et al., 2013). The corpus was used as training data in two CoNLL GEC shared

[1] Section 3.2 presents an extended work initially published in the following papers: Grundkiewicz (2013a) and Grundkiewicz and Junczys-Dowmunt (2014).

[2] http://www.statmt.org/moses/?n=Moses.LinksToCorpora

Corpus	Sentences	Tokens	Annotators
NUCLE	57,151	1,161,567	1
CoNLL-2013 Test Set	1,381	29,207	1
CoNLL-2014 Test Set	1,312	30,144	2
GEC-10 Test Set	1,312	30,144	10

TABLE 3.1: Statistics of the NUCLE error corpora

tasks in 2013 and 2014 (Ng et al., 2014, 2013) and in a number of succeeding works, such as Felice and Yuan (2014), Grundkiewicz and Junczys-Dowmunt (2014), Mizumoto and Matsumoto (2016), Rozovskaya and Roth (2014), Yuan and Briscoe (2016). NUCLE consists of 1,414 essays written by Singaporean students who are non-native speakers of English. The essays cover topics, such as environmental pollution, health care, etc. This corpus includes 57,151 sentences in total. Grammatical errors in these sentences have been manually corrected by professional English teachers and annotated with one of the 27 predefined error type. The error types are presented in Appendix A.

Another 50 essays, collected and annotated similarly as NUCLE, were used in both CoNLL GEC shared tasks (Ng et al., 2014, 2013) as blind test data. The CoNLL-2013 test set has been annotated by one annotator, the CoNLL-2014 by two human annotators. The former data set contains 1,381 sentences, the latter — 1,312. Bryant and Ng (2015) extended the CoNLL-2014 test set with additional annotations from two to ten annotators (we will refer to this resource as GEC-10). Statistics of the NUCLE corpora are presented in Table 3.1; token statistics are provided for the source (erroneous) side.

Other publicly-available ESL learner corpora are: the dataset of FCE scripts (Yannakoudakis et al., 2011)[3] extracted from the proprietary Cambridge Learner Corpus (Nicholls, 2003) (CLC), and the International Corpus Network of Asian Learners of English[4] (ICNALE). They are usually small and do not contain related test sets. NUCLE is a notable exception, but for machine learning approaches even ca. 50,000 sentences form a rather small resource.

There exist a number of publicly available unannotated ESL corpora[5]. However, they have limited application to data-driven approaches to GEC as they do not contain corrections.

3.1.2 Artificial errors

A solution that tries to overcome data sparseness is the creation of artificial data. In the case of artificial error corpora, grammatical errors are introduced by random substitutions, insertions, or deletions. New errors are frequently generated according to the frequency distribution observed in seed corpora.

Izumi et al. (2003) generate artificial errors assuming uniform distribution of target article mistakes made by Japanese ESL learners. Brockett et al. (2006) introduce 14 mass/count noun errors that pose problems to Chinese ESL learners with hand-constructed rules to use as training data for their SMT system. Wagner et al. (2007) produce ungrammatical sentences with four types of errors: context-sensitive spelling errors, agreement errors, errors involving a missing

[3] http://ilexir.co.uk/applications/clc-fce-dataset/
[4] http://language.sakura.ne.jp/icnale/
[5] https://www.uclouvain.be/en-cecl-lcworld.html

word and errors involving an extra word. Error generation was based on an error analysis carried out on a corpus formed by roughly 1,000 error-annotated sentences. Lee and Seneff (2008) artificially introduced verb form errors into a news corpus based on parse trees. A tool for the production of artificial errors that imitate genuine errors has been introduced by Foster and Andersen (2009). Errors are generated using a list of naturally-occurring error patterns (optionally supplied with frequency information), and a grammatically-correct text corpus. Supported error types are word insertion, deletion, substitution or move, and errors at part-of-speech level may be generated. Yuan and Felice (2013) extracted lexical and part-of-speech patterns for five types of errors from NUCLE and applied them to well-formed sentences. The work of Felice and Yuan (2014) extends those experiments by using more linguistic information to derive generation probabilities and create artificial data sets.

Artificial errors can be useful not only for building artificial error corpora, but also for increasing error rate in training data sets to help data-driven methods to spot less frequent errors (Cahill et al., 2013, Rozovskaya and Roth, 2010c).

Researchers have reported different, often contrary, influence of artificial error corpora onto a GEC system performance. Artificial errors can increase the recall at the cost of precision (Rozovskaya and Roth, 2010c) or vice versa (Felice and Yuan, 2014).

Admittedly, artificial error generation is an efficient and economic way to increase the size of training datasets, but the approach has its drawbacks. The diversification of errors in such corpora can be lower due to a small number of real seed data. For specific error types, especially for open-class errors, it may be difficult to create descriptive patterns that can be applied to well-formed sentences (Felice and Yuan, 2014). It is easier to replicate errors within confusion sets, such as articles or prepositions (Rozovskaya and Roth, 2010c). Also, errors involving a redundant words usually require specific methods, for example, detecting spaces preceding noun phrases as potential places where an article or determiner could be incorrectly used. Furthermore, it has been reported that artificial data can be less suited for evaluation purposes (Zesch, 2012).

3.1.3 Text revision histories

An alternative solution for gathering error corpora consists in the extraction of errors from text revision histories. Wikipedia revisions are the most frequently source of data.

Miłkowski (2008) proposes the construction of error corpora from text revision histories based on the hypothesis that the majority of frequent minor edits are the corrections of spelling, grammar, style and usage mistakes. This hypothesis, although accurate, may lead to the situation where some open-class error types, e.g. inflectional errors, are missed. Grundkiewicz (2013a) built a Polish corpus consisting of errors automatically extracted from Wikipedia revisions without the restriction on edit frequency. To distinguish error corrections from unwanted edits and to determine error categories, hand-written rules were used.

Wikipedia revisions have been used for the creation of naturally-occurring corrections and sentence paraphrase corpora (Max and Wisniewski, 2010), evaluation of statistical and knowledge-based measures of contextual fitness for the task of real-word spell checking (Zesch, 2012) and correction of preposition errors (Cahill et al., 2013).

Cahill et al. (2013) confirm that data from Wikipedia is useful for both training a grammatical error correction system and creating artificial data. They show that models trained with Wikipedia data perform well across diversified test sets representing a variety of error distributions. However, the authors have focused only on preposition errors.

Work	Sentences	Tokens
Mizumoto et al. (2011)	391,699	n/a
Yoshimoto et al. (2013)	1,217,124	n/a
Junczys-Dowmunt and Grundkiewicz (2014)	3,733,116	51,259,679
Susanto et al. (2014)	1,114,139	12,945,666
Mizumoto and Matsumoto (2016)	1,069,127	n/a
Yuan and Briscoe (2016)	n/a	28,823,615
Rozovskaya and Roth (2016)	n/a	ca. 48,000,000
Junczys-Dowmunt and Grundkiewicz (2016)	2,186,460	25,732,858

TABLE 3.2: Sizes of the Lang-8 corpus reported in the literature.

The main advantage of Wikipedia-extracted data sets is their size, but there are also disadvantages, for instance Wikipedia's encyclopedic style and an abundance of vandalism.

3.1.4 Social networks for language learners

Probably the best resource for language errors has made a recent appearance in the form of social networks for language learners, an example being *Lang-8*[6]. Learners with different native languages correct one another's texts based on their own native-language skills.

Mizumoto et al. (2011) published a list of learners' corpora[7] that were scraped from the social language learning site Lang-8, called **the Lang-8 Learner Corpora**. Sentences in this data set might be corrected by more than one corrector, have more than one correction or contain additional inline comments, usually added in a less structured manner. Version 1.0 of the Lang-8 corpus is free for academic purposes. Newer versions (2.0) require special license agreements for any usage.

The Lang-8 NAIST corpus has been used by other researchers to build general GEC systems for English as reported in: Mizumoto et al. (2011), Mizumoto and Matsumoto (2016), Rozovskaya and Roth (2016), Susanto et al. (2014), Yoshimoto et al. (2013), Yuan and Briscoe (2016), as well as to create systems tackling specific error types (Cahill et al., 2013, Sawai et al., 2013, Tajiri et al., 2012). The comparison of various versions of the Lang-8 corpus is presented in Table 3.2. Different sizes come from different techniques for filtering noisy sentences. Besides, Mizumoto et al. (2011) made use of sentences written only by Japanese ESL learners. Rozovskaya and Roth (2016) used our Lang-8 WEB corpus that will be described in Section 5.3.1.

Compared to learner error corpora, data from social network services are not annotated in a well-organized manner and thus may contain unrestricted comments or annotations. This, and the fact that the data is automatically processed, causes noise, which has to be taken into account.

3.1.5 Monolingual data

Monolingual data is more widely available and much larger in size than annotated data. In GEC, it is usually used as a source of error-free texts (even though this assumption is usually not true) providing statistics about the correct usage of the language. Common cases of applying monolingual data to GEC are language modeling (Hdez and Calvo, 2014, Junczys-Dowmunt and

[6] http://lang-8.com
[7] http://cl.naist.jp/nldata/lang-8

Grundkiewicz, 2014) and extraction of word probability distribution for ML models (Rozovskaya and Roth, 2014).

One of the most popular source of English data today is the **English version of Wikipedia**[8]. The content of Wikipedia is moderated, mostly by native-speakers, and the article quality is constantly improved, which allows to treat Wikipedia as mostly error-free. On the other hand, the encyclopaedic style differs significantly from the general style of ESL writings.

Another widely used monolingual dataset is **Common Crawl** data[9], a publicly available resource which has been crawled from the web. The main advantage of this resource is its size and the diversity of topics. Crawled texts are of various quality, but it has been shown that web-scale language models can still improve the performance of a number of NLP tasks, such as machine translation (Buck et al., 2014), and grammatical error correction (Junczys-Dowmunt and Grundkiewicz, 2014).

Other large monolingual corpora used for grammatical error correction systems are English Gigaword (Parker et al., 2011), Google Web 1T 5-gram Corpus (Brants and Franz, 2006), or the British National Corpus (BNC)[10].

3.2 The WikEd Error Corpus

In this section, we present the process of building the WikEd Error Corpus: the largest corpus of corrective edits available for the English language.

In contrast to other works that use Wikipedia to build various NLP resources (Cahill et al., 2013, Max and Wisniewski, 2010, Zesch, 2012), we processed the entire English Wikipedia revision history[11] (not only a part of it) and gathered ca. 56 million sentences with annotated edits. The corpus is not limited to specific error types, e.g. real-word spelling errors (Zesch, 2012) or preposition errors (Cahill et al., 2013). WikEd does not exclude specific corrections, such as repetitions or omissions of words, and corrections that refers only to punctuation or case modification as in the work of Max and Wisniewski (2010). Possible application of the created corpus include, but are not limited to, sentence paraphrasing, spelling correction, grammar correction, and more.

The corpus inherits the user-friendly CC BY-SA 3.0 license of the original resource and both, the WikEd Error Corpus and the tools used to produce it have been made available for unrestricted download[12].

3.2.1 Extracting edits from Wikipedia

Wikipedia dumps with complete edit histories are available in the XML format[13]. Similarly to Max and Wisniewski (2010), we iterate over each two adjacent revisions of each Wikipedia page, including articles, user pages, discussions, and help pages. To minimize the number of cases of unwanted vandalism, we skip revisions and preceding revisions if comments contain suggestions of reversions, e.g. *reverting after (. . .)*, *remove vandalism*, *undo vandal's edits*, *delete stupid joke*, etc. This is done by hand-written rules involving regular expressions.

[8] https://en.wikipedia.org
[9] http://commoncrawl.org
[10] http://www.natcorp.ox.ac.uk/corpus
[11] Wikipedia database dump from March 4th, 2014: http://dumps.wikimedia.org/enwiki/20140304/
[12] http://romang.home.amu.edu.pl/wiked/wiked.html
[13] http://dumps.wikimedia.org

Next, we remove XML and Wikitext Markup Language annotations[14] from each article version and split texts into sentences with the NLTK toolkit[15]. Pairs of edited sentences are identified with the Longest Common Subsequence (LCS) algorithm (Maier, 1978) that is applied at the token level. Editions consisting of additions or deletions of full paragraphs are discarded.

Two edited sentences (we will refer to two corresponding edited fragments as sentences, even if they are not well-formed) S_i and S_j are added to the corpus if they meet several surface conditions:

- The sentence length is between 2 and 120 tokens.
- The length difference between S_i and S_j is less than 5 tokens.
- The relative token-based edit distance $d_{\text{edit}}(S_i, S_j)$ with respect to the shorter sentence is smaller than 0.3.

We define the relative token-based edit distance as:

$$d_{\text{edit}}(S_i, S_j) = \frac{d_{LD}(S_i, S_j)\min(|S_i|, |S_j|)}{\log_b \min(|S_i|, |S_j|)},$$

where $d_{LD}(S_i, S_j)$ is the token-based Levenshtein edit distance (Levenshtein, 1966), $|S|$ is the length of the sentence S in tokens. This formula implies that the longer the sentence, the more edits are allowed, but it prevents the acceptance of too many edits for long sentences. The logarithm base b is empirically set to 20.

The threshold values in the above restrictions were chosen experimentally.

3.2.2 Collecting corrective edits

Over 55,109,182 million pairs of edited sentences from the English version of Wikipedia have been collected. The most useful edits include:

- Spelling error corrections:
  You can use rsync to $\left[\underline{\textit{donload}} \rightarrow \textit{download}\right]$ *the database .*,
- Grammatical error corrections:
  There $\left[\underline{\textit{is}} \rightarrow \textit{are}\right]$ *also* $\left[\underline{\textit{a}} \rightarrow \right]$ *two computer games based on the movie .*,
- Stylistic changes:
  $\left[\underline{\textit{Predictably , the}} \rightarrow \textit{The}\right]$ *game ended* $\left[\underline{\textit{predictably}} \rightarrow \right]$ *when she crashed her Escalade. . .*,
- Sentence rewordings and paraphrases:
  These anarchists $\left[\underline{\textit{argue against}} \rightarrow \textit{oppose the}\right]$ *regulation of corporations .*,
- Encyclopaedic style adjustments:
  A $\left[\underline{\textit{local education authority}} \rightarrow \textit{Local Education Authority}\right]$ *(LEA) is the part of a council in England or Wales.*

The WikEd corpus contains also less useful edits for grammatical error correction task, e.g.:

[14] http://medialab.di.unipi.it/wiki/Wikipedia_Extractor
[15] http://nltk.org

Edits	Freq.	Edits	Freq.	Edits	Freq.
`ins(«,»)`	1,169,076	`ins(«and»)`	65,867	`ins(«in»)`	36,890
`del(«,»)`	501,149	`del(«a»)`	62,920	`sub(«,»,«and»)`	36,799
`ins(«the»)`	288,175	`del(«"»)`	57,276	`sub(«and»,«,»)`	36,240
`del(«the»)`	249,101	`del(«also»)`	56,943	`sub(«an»,«a»)`	35,766
`sub(«is»,«was»)`	158,221	`del(«and»)`	56,687	`ins(«to»)`	35,534
`ins(«a»)`	93,019	`sub(«a»,«an»)`	54,917	`del(«in»)`	30,648
`sub(«are»,«were»)`	79,435	`sub(«was»,«is»)`	48,729	`del(«has»)`	29,778
`del(«of»)`	79,310	`ins(«.»)`	47,030	`ins(«is»)`	29,012
`ins(«"»)`	77,788	`ins(«also»)`	46,087	`sub(«the»,«a»)`	28,301
`sub(«it 's»,«its»)`	72,419	`ins(«of»)`	45,609	`ins(«'s»)`	28,241

TABLE 3.3: The most frequent edits in the WikEd corpus.

- Time reference changes:
  The Kiwi Party $[\underline{is} \rightarrow was]$ *a New Zealand political party formed in 2007 .*,
- Information supplements:
  Aphrodite is the Greek goddess of love $[\ \rightarrow\ ,\ sex]$ *and beauty .*,
- Numeric information updates:
  In $[\underline{May\ 2003} \rightarrow August\ 2004]$ *this percentage increased to* $[\underline{62} \rightarrow 67]$ *% .*,
- Item additions/deletions to/from bulleted lists:
  Famous Bronxites include $[\ \rightarrow Regis\ Philbin\ ,]$ *Carl Reiner , Danny Aiello. . . ,*
- Amendments of broken MediaWiki's markups:
  The bipyramids are the $[\underline{[\ [\ dual\ polyhedron\ /} \rightarrow\ ]$ *dual polyhedra* $[\underline{[\ [} \rightarrow\ ]$ *of the prisms .*,
- Changes made by vandals:
  David Zuckerman is a writer and $[\underline{producer} \rightarrow poopface]$ *for television shows.*

The total number of edits is 71,474,188, among which 11,920,778 (16.68%) are deletions and 17,711,567 (24.78%) insertions. The most frequently occurring edits are presented in Table 3.3. The `sub(·,·)` stands for word(s) substitutions, `del(·)` for deletions, and `ins(·)` for insertions.

3.2.3 Edition filtering

Sentences with potentially unwanted edits, e.g. updates of bulleted list, amendments of MediaWiki markup, and vandalism can be effectively filtered out using simple heuristic rules. For example, all pairs of sentences S_i and S_j that satisfy the following conditions can be discarded:

- Either sentence S_i or S_j contains a vulgar word (determined by the list of vulgarisms) or a very long sequence of character with no spaces (e.g. produced by random keystrokes).
- Any of the sentences S_i or S_j contains fragments of XML or Wiki markup, e.g. `<ref>`, `<br>` or `[http:`.
- Edits concern changes in dates or numerical values only.
- The only edit consists of removing a full stop or semicolon at the end of the sentence S_i.

- The ratio of non-word tokens in S_j to word tokens is higher than a given threshold (we used 0.5).

In the version of the WikEd corpus which has been made publicly available, these sentences are only marked as potentially harmful, but not removed. For instance, vandalized entries may be useful for various tasks by themselves.

3.2.4 Corpus format

It was our intention to release the WikEd Error Corpus in a machine-friendly format. We chose a representation based on GNU `wdiff` output[16] extended by comments including meta-data. For example, for a sentence *This page lists links about ancient philosophy.* with the following two edits: insertion of *some* at third position and substitution of *about* with *to*, the WikEd entry corresponds to:

`This page lists {+some+} links [-about-] {+to+} ancient philosophy`.

Meta-data consist of:

- The revision id, accompanying comment, and revision timestamp.
- The title and id of the edited Wikipedia page.
- The name of the contributor or IP address in case of an anonymous edition.

All sentences preserve the chronological order of the original revisions.

3.3 Summary

Both error corpora and monolingual data are crucial to create effective grammatical error correction systems using data-driven approaches. In this chapter we described the types of error corpora used in automated ESL grammatical error correction: annotated ESL learner writings, artificially generated errors, corrections scraped from the social network sites, and edits extracted from text revisions. We discussed their advantages and disadvantages. The sources of large-scale monolingual data have also been shown.

We also introduced the WikEd Error Corpus — a new large resource with possible applications to sentence paraphrasing, spelling correction, and — as will be showed in Chapter 5 — grammatical error correction.

[16] https://www.gnu.org/software/wdiff/manual/wdiff.html#wdiff

Chapter 4

Evaluation Metrics

For a fair comparison of two GEC systems, we need both shared datasets and an evaluation framework. It is almost impossible to compare systems evaluated on different test sets or against different evaluation metrics (Chodorow et al., 2012). Automatic evaluation, much easier to perform than human evaluation, allows for fast, reproducible and objective feedback on system performance during development and testing (Dahlmeier and Ng, 2012b). The choice of the evaluation metric is important due to the fact that GEC systems, and other NLP systems, should be developed and tuned according to a given metric in order to maximize the overall performance. The latter is often neglected.

In this chapter, we discuss difficulties in the evaluation of grammatical error correction systems in Section 4.1. Next, in Section 4.2, we introduce commonly-used evaluation metrics including standard metrics originated from information retrieval, metrics developed specifically for GEC, and metrics introduced for machine translation. In order to find the optimal metric for the task, we evaluate selected metrics in terms of correlation with human judgment by conducting the first large-scale human evaluation study of GEC systems presented in Section 4.3[1]. We describe the collection of human judgments, which are then turned into ordered rankings, and analyze their correlation with standard automatic metrics used in GEC.

4.1 Difficulties in evaluating GEC systems

Chodorow et al. (2012) draw attention to a number of evaluation issues in error correction, which make it hard to compare different approaches. Lack of consensus in evaluation methods is mainly due to the nature of the task. Main difficulties come from the following phenomena:

- The highly skewed distribution of positive and negative classes, i.e. the low frequency of errors as compared to correctly used words.

  For example, the baseline accuracy computed at the token level is very high as the WER in ESL data is usually low (see Table 5.6). Examining the performance of a GEC system on such a high accuracy above 90–95% can be misleading (Chodorow et al., 2012). According to Chodorow et al. (2012), it is advisable to use automatic evaluation metrics, which take into account the fact that the majority part is the negative class (i.e. non-errors), such as F-score (Rijsbergen, 1979) or the Cohen's κ coefficient (Cohen, 1960a).

[1] Section 4.3 presents an extended work originally published in Grundkiewicz et al. (2015).

- Multiple possible corrections for a given error.

  This poses two potential issues: firstly, some of the corrections may be optional and others mandatory, and it is not obvious whether they should be of the same importance; secondly, the gold standard may not contain all the edits that may or should be made. A solution presented by Madnani et al. (2011) consists in weighting errors based on the distribution of several annotations for each error and use versions of the evaluation metrics which can use these weights. This requires the existence of datasets with multiple annotations, which are usually very small and costly to build, such as the extended version of the CoNLL-2014 test set created by Bryant and Ng (2015).

- A non-unique set of edits that transforms one string into another, even if the token level is chosen as the evaluation unit size.

  The span of the edit between the input sentence and the correction candidate for a given error can mismatch the extent of the annotation in the reference correction. For instance, for a sentence *There is no a doubt.*, a hypothetical system may propose an edit $a \rightarrow \varnothing$[2], whereas the gold standard includes the annotation *a doubt* $\rightarrow$ *doubt*. Such a mismatch results in the underestimation of the system performance. As pointed out by Sakaguchi et al. (2016), this issue has to be considered not only during the development of an evaluation framework, but also at the time of designing a data annotation schema.

- The size of the negative class (i.e. non-errors) being difficult to determine.

  The negative class instances are not obvious to determine as this requires the identification of all positions in a sentence where an error can occur. For example, in the case of article or determiner errors, it is not evident which positions in a sentence should be counted as instances of the zero article $\varnothing$. The more instances of $\varnothing \rightarrow \varnothing$ edits, the higher the number of true negatives and the higher the accuracy (Chodorow et al., 2012).

- Interdependence of errors.

  Some errors are interdependent, which means that one change in a sentence may require another change. It is not obvious how strict the evaluation framework should be in scoring partial corrections within interdependent errors.

No single metric serves all purposes. Chodorow et al. (2012) argue that the choice of the metric should take into account the type of application that the system is used for. A measure that is useful for system comparison may not be the best to determine if a system is good enough to be deployed operationally, and measures of overall sentence quality, may not adequately support language learning. When building a Computer-Assisted Language Learning (CALL) system is a goal, fluency may be favoured instead of technical grammaticality encoded in annotated error corpora (Sakaguchi et al., 2016). Similarly, for a proofreader, recall may be favoured at the expense of precision, and vice versa for a user interested in fast and risk-free automatic improvement of the document quality.

[2] The symbol $\varnothing$ stands for the zero article.

	Input	Reference	Candidate
TN	a	a	a
FP	a	a	b
FN	a	b	a
TP	a	b	b
*	a	b	c

TABLE 4.1: WAS evaluation scheme.

4.2 Evaluation metrics

Most of the automatic evaluation metrics used for GEC take into account the relation between the elements of the triples consisting of the input (or source) sentence S, the system's output C (the candidate correction or the hypothesis), and one or more annotator's references R (a gold standard or an annotator correction).

Below, we introduce the most popular evaluation metrics specific for GEC and two of the most popular metrics from machine translation.

4.2.1 Standard metrics

Automatic metrics calculated against gold standard annotations are based on a *contingency table*, which compares system edits with reference edits. Each system edit can be classified into one of four types:

- A true positive (TP) is a "hit" indicating a match between the system edit and the reference edit.
- A true negative (TN) is equivalent with a correct rejection, i.e. the system does not propose an edit and it is not present in the reference.
- A false positive (FP) is a "false alarm" (or type I error), which indicates that the system introduces an unnecessary edit which is not present in the reference.
- A false negative (FN) is a "miss" (or type II error) indicating a correction missed by the system.

Chodorow et al. (2012) proposed a three-way contingency table called Writer-Annotator-System (WAS) evaluation scheme where tree edits (from the input, reference and hypothesis) are compared to determine the classification type. The WAS evaluation scheme is presented in Table 4.1. Depending on whether we are interested in measuring the system performance in the detection or correction task, the star in the WAS evaluation scheme represents a different type. For error detection the case where three edits differ from each other is a TP, for error correction it is both FP and FN.

Using this notation, the measures of *accuracy* (Acc), *precision* (P) and *recall* (R) are formulated as follows:

$$Acc = \frac{TP + TN}{N},$$

$$P = \frac{TP}{TP + FP},$$
$$R = \frac{TP}{TP + FN}.$$

Accuracy measures the ratio of correct decisions the system made to all possible decisions. P is the proportion of properly corrected errors out of all proposed corrections. The lower the number of reported false alarms, the higher the precision. R, in turn, reports the proportion of properly corrected errors among all errors annotated in the reference. Recall is sometimes referred to as a measure of coverage.

A commonly used measure that combines P and R is *F-score* (or F-measure) (Rijsbergen, 1979). It is defined as the harmonic mean of the two:

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

A positive parameter β determines the trade-off between P and R. The most popular F-scores used in GEC are F_1 and $F_{0.5}$. F_1 weights precision and recall equally, whereas $F_{0.5}$ weights precision twice as much as recall.

The range of F_β-score is between 0 and 1. A score of 1 requires that all reference edits have to be matched with system edits without false alarms, i.e. both P and R have to be equal to 1.

4.2.2 MaxMatch

The *MaxMatch* (M^2) (Dahlmeier and Ng, 2012b) evaluation metric addresses the problem of edits mismatched due to non-identical scopes of the system and reference edits. For example, given the exemplary source sentence *There is no a doubt.* and the reference edit created by an annotator *a doubt* → *doubt*, the accurate correction proposed by a GEC system *There is no doubt.* might still not match the reference if the system edit is extracted as a longest common subsequence between the source and system output, i.e. *a* → ∅.

The algorithm computes set of phrase-level system edits with the maximum overlap with the reference by finding the shortest path in the edit lattice graph. The graph is built based on the token-level Levenshtein distances between a source sentence and hypothesis (hence, M^2 is a sentence-level metric). A given set of candidate edits $c_i, \ldots, c_n$ is subsequently scored with respect to the reference edits $r_i, \ldots, r_m$ with F_β-score where P and R are defined as follows:

$$P = \frac{\sum_{i=1}^{n} |c_i \cap r_i|}{\sum_{i=1}^{n} |c_i|},$$
$$R = \frac{\sum_{i=1}^{n} |c_i \cap r_i|}{\sum_{i=1}^{n} |r_i|}.$$

The M^2 metric allows for multiple reference annotations and chooses references that maximize the overall F-score. Hence, the intersection between c_i and r_i is defined as:

$$c_i \cap r_i = \{c \in c_i \,|\, \exists r \in r_i \,:\, \text{match}(c, r)\},$$

where $\text{match}(c, r)$ determines whether the system edit c matches the reference edit r and depends on the chosen task.

Due to the adoption of M^2 as the main evaluation metric during the CoNLL shared tasks on GEC and the QALB shared task on automatic text correction for Arabic (Mohit et al., 2014b), the metric has become the *de facto* standard metric for GEC.

However, the M^2 metric is not perfect. Its results depend heavily on various hyperparameters, i.e. the number of unchanged words than can appear in an edit or the handling of casing and whitespace. M^2 results are heavily influenced by the choice of β in the F_β-score formula. Moreover, M^2 completely fails to score system output which can be judged to be worse that the untouched input. The input always gets a score of 0. A higher score is assigned to a system output which corrects at least one error from the gold standard regardless of how many new errors are introduced.

Between the CoNLL-2013 and CoNLL-2014 shared tasks, the organizers changed β from 1.0 to 0.5, and motivated the change with intuition alone. The QALB shared tasks for Arabic continue to use $\beta = 1.0$ — setting $\beta = 0.5$ would have changed the outcome of these tasks.

4.2.3 I-WAcc

The *I-WAcc* (I-measure on Weighted Accuracy) metric or group of metrics proposed by Felice and Briscoe (2015) tries to address some of the shortcomings of M^2. The metric is defined as follows:

$$\text{I-WAcc} = \begin{cases} \lfloor \text{WAcc}_c \rfloor & \text{if } \text{WAcc}_c = \text{WAcc}_s \\ \frac{\text{WAcc}_c - \text{WAcc}_s}{1 - \text{WAcc}_s} & \text{if } \text{WAcc}_c > \text{WAcc}_s \\ \frac{\text{WAcc}_c}{\text{WAcc}_s} - 1 & \text{otherwise} \end{cases}.$$

WAcc_c and WAcc_s are Weighted Accuracies calculated for the system output and the input text respectively. WAcc is defined as:

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

where w is a weight which rewards corrections over preservation and simultaneously penalizes false alarms over missed corrections. FPN stands for cases where source, candidate and reference edit texts differ from each other (the last row in Table 4.1).

Felice and Briscoe (2015) propose to use WAcc claiming that the accuracy seems a more appropriate measure of text quality than F-score because it takes true negatives into consideration. The inclusion of true negatives into the calculations makes this a very conservative metric, quite similar to the machine translation metrics described hereafter.

The I-measure part is an idea that can be treated separately from WAcc. It assigns negative weights to systems that are harmful with regard to the input text. The metric returns values from the range $[-1, 1]$. A system equivalent to the input receives a score of 0, system output that are worse than the input are assigned negative scores.

4.2.4 MT metrics

Using statistical machine translation for building grammatical error correction systems inspired researchers to use machine translation evaluation metrics in the GEC field.

The most popular MT evaluation metric is **BLEU** (BiLingual Evaluation Understudy) (Papineni et al., 2002). BLEU is defined as an n-gram match precision between a candidate sentence C and a reference sentence R with a length penalty:

$$\text{BLEU}(C,R) = \exp\big(\sum_{1}^{n} w_n \log P_n(C,R)\big) \cdot Pen_{\text{BLEU}}(\cdot),$$

where n is a maximum length of n-gram (usually set to 4 as it has been shown to have the highest correlation with human judgments); w_n are positive weights; $P_n(C,R)$ is a modified precision score calculated as a sum over the matches for each n-gram length for every candidate sentence S in entire dataset; and $Pen_{\text{BLEU}}(\cdot)$ is a brevity penalty (Papineni et al., 2002), which prevents short candidates from receiving too high a score.

BLEU is calculated at corpus level instead of sentence level. Its output is in the range of $[0,1]$. The higher the value, the more similar a hypothesis is to the reference. The value 1 is attained by the candidate texts that are identical to reference texts. Therefore, in the case of grammatical error correction evaluation, BLEU values are usually very high.

Another popular MT metric is **METEOR** (Denkowski and Lavie, 2011). It solves several issues found for BLEU and has unique features, such as stemming and synonymy matching. The METEOR metric is a tunable metric based on the harmonic mean of P and R calculated for unigrams:

$$\text{METEOR}(C,R) = \frac{P_1(C,R)R_1(C,R)}{\alpha R_1(C,R) + (1-\alpha)P_1(C,R)} \cdot \big(1 - Pen_{\text{METEOR}}(\cdot)\big).$$

$Pen_{\text{METEOR}}(\cdot)$ is a fragmentation penalty, which helps to address gaps and differences in word order.

Both BLEU and METEOR can be computed on several references.

4.3 Human evaluation of GEC systems

Machine translation researchers are aware of the fact that automatic metrics of MT quality are an imperfect substitute for human assessments. Therefore manual evaluation of the system outputs are regularly conducted — the best known are campaigns organized during the annual Workshops on Statistical Machine Translation (WMT)[3]. The human evaluation campaigns are an important driving factor for the progress in the field. Their results are reported as the final rankings of MT shared tasks and used to evaluate automatic metrics or analyze human perception of MT quality.

We conducted the first large-scale human evaluation of automatic grammatical error correction systems inspired by similar efforts for machine translation, most notably WMT campaigns (Bojar et al., 2013, 2014, Macháček and Bojar, 2014a). We adapted the methods developed for recent iterations of the WMTs to GEC and modified them where necessary. The processes of collecting human judgments, computing new human-based system rankings and measuring their correlation with selected GEC metrics[4] are described in the subsequent sections.

[3] http://www.statmt.org/wmt16/

[4] The collected data in form of rankings and pairwise judgments and the tools used to calculate rankings, clusters, inter-annotator agreement and metric correlations are available for download from the repository: https://github.com/grammatical/evaluation

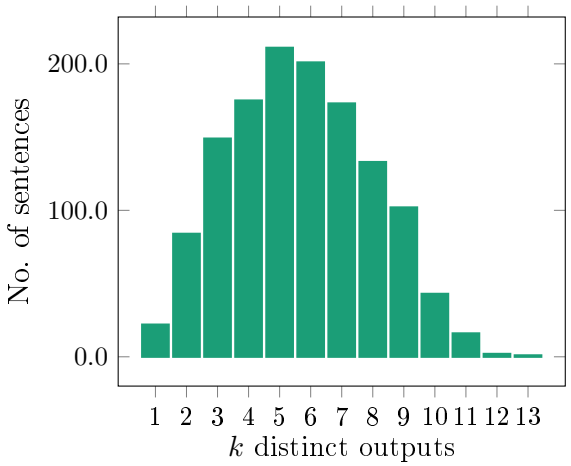


FIGURE 4.1: Frequencies of distinct corrected sentences per input sentence for systems participating in the CoNLL-2014 shared task.

4.3.1 Data collection

We used the publicly available system outputs of the CoNLL-2014 shared task as evaluation data. The test set consists of $I = 1312$ sentences. Twelve system outputs are available and each output consists of the corrected versions of all sentences from the test set[5].

In GEC evaluation, contrary to MT evaluation, the input has to be also considered. Often system outputs are equal to the unmodified input, as it is most desirable if there are in fact no errors. We therefore included the INPUT as a separate system. This finally results in $N = 13$ system outputs that need to be judged.

4.3.1.1 Sampling sentences for evaluation

Due to the relatively small number of modifications that GEC systems apply to the input there is not only a large overlap with the input, but also among outcomes of compared systems. Figure 4.1 illustrates the number of test sentences for which the 13 systems produced k distinct outputs. There is only one case where all systems proposed different corrections[6] and there are 22 sentences where all outputs were the same[7]. On average there are 5.7 distinct proposed corrections per a test sentence. This forms a need for the efficient collection of pairwise judgments.

Following the WMT human evaluations (Bojar et al., 2013, 2014), we decided to present up to $M = 5$ unique system outputs to be judged by the annotator in a single ranking. Judging too many sentences at once would prolong the evaluation time for a single input sentence, what we wanted to avoid. Assuming that sentences are sampled uniformly, we can either select five candidates among the 13 different systems or select at most five candidates from distinct outputs, recording which systems overlapped. In both cases there are drawbacks:

- In the first case, if we sample systems uniformly, we lose easily obtainable pairwise judgments for systems with the same output.
- In the latter case, if we collapse before sampling, we introduce a strong bias towards ties.

[5]The thirteenth participant NARA is missing from this set as this team did not submit a system description.
[6]Actually, an incorrectly joined long segment consisting of several sentences.
[7]Since the input is included, i.e. equal to the input.

To counter the bias from the second method, we abandon uniform sampling of test set sentences and use instead a parametrized distribution that favors more diverse, hence more informative, sets of outputs. Below, we present our method for calculating the probability of sampling a set of outputs.

Let us recall that I (= 1312) is the number of sentences in the test set, N (= 13) is the number of systems to be evaluated, M is the maximum number of sentences presented to the evaluator in a single ranking (we use $M = 5$), furthermore $M \leq N$. As each system produced a single output for each test sentence, the set of system outputs to be evaluated $E = \{O_1, \ldots, O_I\}$ consists of I sets O_i of N output sentences each, i.e. $\forall_{1 \leq i \leq I} |O_i| = N$. Each sentence in O_i can overlap with other sentences multiple times. Thus, for each set O_i we define the corresponding multiset of multiplicities U_i, such that $\sum_{u \in U_i} u = N$.

Example 4.1 (Set of outputs and multiset of multiplicities)**.** *Let assume $N = 13$, i.e. there are 13 system outputs. For an i-th sentence, three systems generate an identical output* `A`*, five systems generate an output* `B`*, two systems generate an output* `C`*, and the remaining systems generate different outputs* `D`*,* `E` *and* `F`*. In this case, the set $O_i = \{$*`A`$_1$*,* `A`$_2$*,* `A`$_3$*,* `B`$_1$*,* `B`$_2$*,* `B`$_3$*,* `B`$_4$*,* `B`$_5$*,* `C`$_1$*,* `C`$_2$*,* `D`*,* `E`*,* `F`$\}$*. The corresponding multiset of multiplicities is $U_i = \{3, 5, 2, 1, 1, 1\}$.*

We define $c_i(j)$ as the number of possible ways to choose at most M different sentences that cover j systems for the i-th set of outputs:

$$c_i(j) = \left| \left\{ S \subseteq U_i : |S| \leq M \wedge \sum_{u \in S} u = j \right\} \right|.$$

Then the expected number C_i of systems covered by choosing at most M sentences is:

$$C_i = \frac{\sum_{j=M}^{N} c_i(j) \cdot j}{\sum_{j=M}^{N} c_i(j)}.$$

The pseudo-probability p'_i of sampling the i-th sentence is defined as:

$$p'_i = \frac{\binom{M}{2}}{\binom{C_i}{2}}, \quad \text{where } \binom{C_i}{2} = \frac{C_i(C_i - 1)}{2},$$

which is the ratio of pairwise comparisons of M versus C_i different systems.

Example 4.2 (Pseudo-probability)**.** *Let $N = 13$ and $M = 5$, i.e. there are 13 system outputs and no more than 5 sentences are presented to human evaluator. For the multiset of multiplicities $U_i = \{3, 5, 2, 1, 1, 1\}$ as presented in Example 4.1, we get the following values of $c_i(j)$ for $j = 5 \ldots 13$:*

$$c_i(5) = |\{\{\texttt{A}, \texttt{C}\}, \{\texttt{A}, \texttt{D}, \texttt{F}\}, \{\texttt{A}, \texttt{E}, \texttt{D}\}, \{\texttt{A}, \texttt{E}, \texttt{F}\}, \{\texttt{B}\}, \{\texttt{C}, \texttt{E}, \texttt{D}, \texttt{F}\}\}| = 6$$
$$c_i(6) = |\{\{\texttt{A}, \texttt{C}, \texttt{D}\}, \{\texttt{A}, \texttt{C}, \texttt{E}\}, \{\texttt{A}, \texttt{C}, \texttt{F}\}, \{\texttt{A}, \texttt{E}, \texttt{D}, \texttt{F}\}, \{\texttt{B}, \texttt{D}\}, \{\texttt{B}, \texttt{E}\}, \{\texttt{B}, \texttt{F}\}\}| = 7$$
$$c_i(7) = |\{\{\texttt{A}, \texttt{C}, \texttt{D}, \texttt{F}\}, \{\texttt{A}, \texttt{C}, \texttt{E}, \texttt{D}\}, \{\texttt{A}, \texttt{C}, \texttt{E}, \texttt{F}\}, \{\texttt{B}, \texttt{D}, \texttt{F}\}, \{\texttt{B}, \texttt{E}, \texttt{D}\}, \{\texttt{B}, \texttt{E}, \texttt{F}\}, \{\texttt{C}, \texttt{B}\}\}| = 7$$

$$
\begin{aligned}
c_i(8) &= |\{\{A,B\},\{A,C,E,D,F\},\{B,E,D,F\},\{C,B,D\},\{C,B,E\},\{C,B,F\}\}| = 6\\
c_i(9) &= |\{\{A,B,D\},\{A,B,E\},\{A,B,F\},\{C,B,D,F\},\{C,B,E,D\},\{C,B,E,F\}\}| = 6\\
c_i(10) &= |\{\{A,B,D,F\},\{A,B,E,D\},\{A,B,E,F\},\{A,C,B\},\{C,B,E,D,F\}\}| = 5\\
c_i(11) &= |\{\{A,B,E,D,F\},\{A,C,B,D\},\{A,C,B,E\},\{A,C,B,F\}\}| = 4\\
c_i(12) &= |\{\{A,C,B,D,F\},\{A,C,B,E,D\},\{A,C,B,E,F\}\}| = 3\\
c_i(13) &= |\emptyset| = 0
\end{aligned}
$$

These give the expected number of systems covered by choosing at most 5 *outputs* $C_i = 354/44 \approx 8.02$ *and the pseudo-probability* $p'_i \approx \binom{5}{2}/\binom{8.02}{2} = 10/28.17 \approx 0.35$

Example 4.3 (Pseudo-probability for unique outputs)**.** *Let* $N = 13$ *and* $M = 5$*, and consider the extreme case where each system generates a different output. For this example, the set* O_i *consists of 13 different systems outputs and the corresponding multiset of multiplicities consists of 13 times 1, i.e.* $U_i = \{1,1,1,1,1,1,1,1,1,1,1,1,1\}$*.* $C_i = 5$*, since* $c_i(j)$ *is equal to 1 for* $j = 5$ *and 0 for all* $j \neq 5$*, since there is no subset of* U *that has 5 or less elements and sums to more than 5. The pseudo-probability* $p'_i = \binom{5}{2}/\binom{5}{2} = 1$*.*

In other words, for a set of all different outputs we will always be able to cover 5 different systems if we choose 5 sentences, hence $p'_i = 1$*.*

Example 4.4 (Pseudo-probability for identical outputs)**.** *Let again* $N = 13$ *and* $M = 5$*. For the extreme case when all systems generate the same output,* $U_i = \{13\}$*. Since choosing a sentence covers automatically all systems,* $c_i(j) = 1$ *for* $j = 13$ *and* $c_i(j) = 0$ *for* $j \neq 13$*. In that case* $C_i = 13$ *and* $p'_i = \binom{5}{2}/\binom{13}{2} = 10/78 \approx 0.13$*.*

The final probability p_i of sampling the i-th set of outputs can be obtained by normalizing over the entire set of output sets:

$$
p_i = \frac{p'_i}{\sum_{j=1}^{|E|} p'_j} = \frac{\binom{C_i}{2}^{-1}}{\sum_{j=1}^{|E|} \binom{C_j}{2}^{-1}}.
$$

4.3.1.2 Collecting system rankings

To collect human judgments we used the open-source evaluation system Appraise (Federmann, 2010), which is also used annually to run human evaluations of WMT.

Similarly as in the WMT human evaluation tasks, annotators were asked to rank sentences from best to worse. Ties were allowed. Judges were aware that the absolute ranks bear no relevance as ranks are later turned into relative pairwise judgments. No notion of "better" or "worse" was imposed by the authors, we relied on the judges to develop their own intuition. All judges were English native speakers and have strong backgrounds in linguistics, natural language processing, or teaching English to ESL learners.

Several modifications to the Appraise framework were implemented[8] to account for the specific nature of GEC and our sampling method. These changes include:

[8] The fork of the original source code which contains modifications for GEC can be found at https://github.com/snukky/Appraise

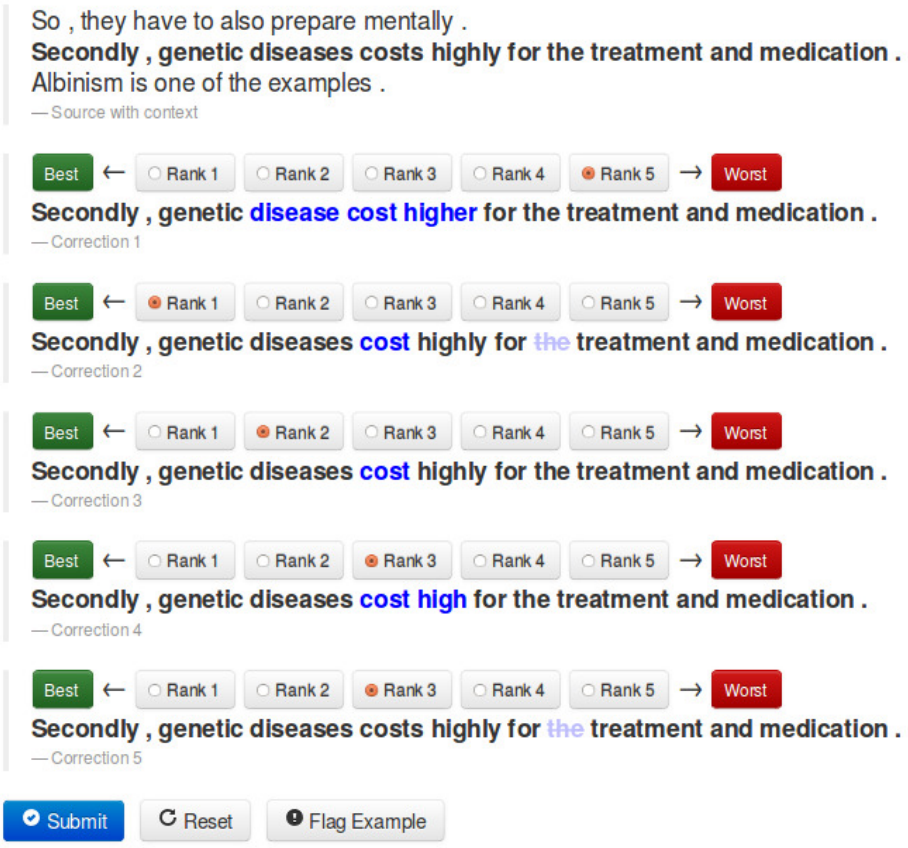


FIGURE 4.2: Screenshot of Appraise modified for GEC judgment.

- Only the input sentence is displayed (top, bold), and no reference correction is given.
- The input sentence is surrounded by one preceding and one following sentence to illustrate context.
- Identical corrections are collapsed into a single output, and all system names with the same output are recorded internally.
- Edited fragments are highlighted: blue for insertions and substitutions, pale blue and crossed-out for deletions.

By highlighting edited fragments, we aim to facilitate the identification of differences between proposed corrections. Due to the high similarity between candidates, these differences might have been easily missed. Figure 4.2 displays a screenshot of our modified version of Appraise with a sentence under evaluation.

We collected system rankings from eight judges. Each judge ranked from 70 to 400 system outputs, which resulted in the total 2,319 collected ranks (first two columns in Table 4.2).

4.3.1.3 Pairwise judgments

We turned collected individual rankings into sets of relative pairwise judgments, where the lower ranked system scores a win.

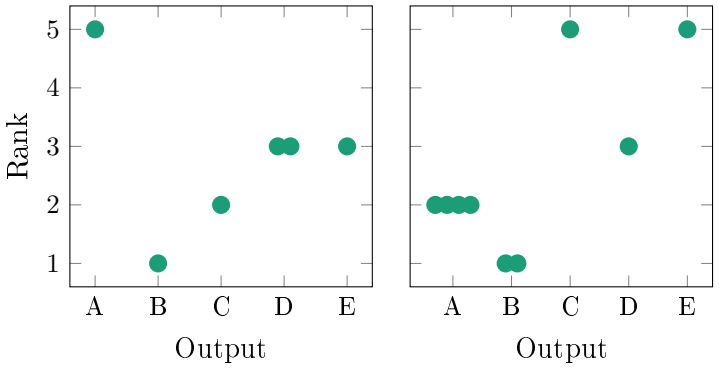


FIGURE 4.3: Two authentic example rankings with overlapping system outputs.

Figure 4.3 presents two example rankings with differing degrees of overlap between systems. The first ranking (left) corresponds to Figure 4.2 and contains five outputs from six systems. Output `D` is produced by two different systems, outputs `A`, `B`, `C` and `E` are unique. A human evaluator judged the system output `B` as best (rank 1), output `C` as second (rank 2), outputs `D` and `E` are judged equivalently as third (rank 3), and `A` is judged to be worst (rank 5). The second ranking (right) presents five unique outputs from nine GEC systems: output `A` was produced by four systems, output `B` by two, and outputs `C`, `D` and `E` were unique.

When converting an individual ranking into a set of pairwise judgments of the form `A>B`, `A=B`, `A<B`, absolute ranks and differences are lost[9]. Pairs from within overlaps result in ties, e.g. `D=E` for the first ranking in Figure 4.3. Pairs between overlaps are expanded as full products, e.g. `A>D` results in $\texttt{A>D}_1$ and $\texttt{A>D}_2$.

Due to the collapsing of identical outputs we obtained significantly more data than the usual 10 pairs from one ranking with five sentences. For example, the following $\binom{6}{2} = 15$ pairwise judgments can be obtained from the left ranking in Figure 4.3:

$\texttt{A>B A>C A>D}_1\ \texttt{A>D}_2\ \texttt{A>E}$
$\texttt{B<C B<D}_1\ \texttt{B<D}_2\ \texttt{B<E}$
$\texttt{C<D}_1\ \texttt{C<D}_2\ \texttt{C<E}$
$\texttt{D}_1\texttt{=D}_2\ \texttt{D}_1\texttt{=E}\ \texttt{D}_2\texttt{=E}$

Greater overlap results in more pairwise judgments for the second example ($\binom{9}{2} = 36$):

$\texttt{A}_1\texttt{=A}_2\ \texttt{A}_1\texttt{=A}_3\ \texttt{A}_1\texttt{=A}_4\ \texttt{A}_1\texttt{>B}_1\ \texttt{A}_1\texttt{>B}_2\ \texttt{A}_1\texttt{<C}\ \texttt{A}_1\texttt{<D}\ \texttt{A}_1\texttt{<E}$
$\texttt{A}_2\texttt{=A}_3\ \texttt{A}_2\texttt{=A}_4\ \texttt{A}_2\texttt{>B}_1\ \texttt{A}_2\texttt{>B}_2\ \texttt{A}_2\texttt{<C}\ \texttt{A}_2\texttt{<D}\ \texttt{A}_2\texttt{<E}$
$\texttt{A}_3\texttt{=A}_4\ \texttt{A}_3\texttt{>B}_1\ \texttt{A}_3\texttt{>B}_2\ \texttt{A}_3\texttt{<C}\ \texttt{A}_3\texttt{<D}\ \texttt{A}_3\texttt{<E}$
$\texttt{A}_4\texttt{>B}_1\ \texttt{A}_4\texttt{>B}_2\ \texttt{A}_4\texttt{<C}\ \texttt{A}_4\texttt{<D}\ \texttt{A}_4\texttt{<E}$
$\texttt{B}_1\texttt{=B}_2\ \texttt{B}_1\texttt{<C}\ \texttt{B}_1\texttt{<D}\ \texttt{B}_1\texttt{<E}\ \texttt{B}_2\texttt{<C}\ \texttt{B}_2\texttt{<D}\ \texttt{B}_2\texttt{<E}$
`C>D C=E`
`D<E`

Table 4.2 lists the full statistics for collected rankings by individual annotators. Unexpanded pairs are WMT-style pairwise judgments before an output `A` gets split into overlapping systems $\texttt{A}_1$, $\texttt{A}_2$, $\texttt{A}_3$, etc. Ties for unexpanded pairs stem from equally ranked different outputs. The large number of ties for expanded pairs is to be expected due to the high overlap between systems.

[9]For example, a ranking similar to the left ranking from Figure 4.3 where the system output `A` is ranked as fourth instead of fifth would be converted to the same set of pairwise judgements.

Judge	Ranks	Unexpanded	Expanded
1	400	3525 (1022)	18400 (10166)
2	299	2684 (1099)	13657 (8429)
3	400	3523 (914)	18912 (9684)
4	201	1750 (550)	9478 (5539)
5	349	3099 (766)	17107 (8972)
6	400	3474 (517)	19313 (9209)
7	70	646 (145)	3383 (1593)
8	200	1815 (681)	8848 (5525)
Total	2319	20516 (5694)	109098 (59117)

TABLE 4.2: Statistics for collected rankings (Ranks), unexpanded pairwise judgments and expanded pairwise judgments. Corresponding numbers for ties are given in parentheses.

4.3.1.4 Inter- and intra-annotator agreement

Again inspired by the WMT evaluation campaigns, we computed annotator agreement as a measure of reliability of the pairwise judgments with the Cohen's κ coefficient (Cohen, 1960b):

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)},$$

where $P(A)$ is the proportion of times that the annotators agree, and $P(E)$ is the proportion of times that they would agree by chance. κ assumes values from 0 (no agreement) to 1 (perfect agreement).

All probabilities were computed as ratios of empirically counted pairwise judgments. As judges worked on collapsed outputs, we calculated agreement scores for unexpanded pairs. Otherwise, the high overlap would unfairly increase agreement. $P(A)$ was calculated by examining all pairs of outputs, which had been judged by two or more judges, and counting the proportion of cases when the judges on the comparison agreed that `A<B`, `A=B`, or `A>B`. $P(E) = P(\texttt{A<B})^2 + P(\texttt{A=B})^2 + P(\texttt{A>B})^2$ is the probability that two judges agree randomly. Intra-annotator agreement as a measure of consistency was calculated for output sets that had been judged more than once by the same annotator.

Agreement numbers in Table 4.3 are in the lower range of values reported for MT human evaluation tasks organized during WMT (Bojar et al., 2014). However, it should be noted that judges never see repeated outputs within one ranking which probably decreases agreement compared to the MT-specific task. While sentence-wise agreement is rather weak, we see in the Subsection 4.3.3 that global system rankings computed for individual judges are highly correlated.

4.3.2 Computing ranks

In this section, it is our aim to produce a system ranking from best to worse by computing the average number of cases each system was judged better than other systems based on the collected pairwise rankings. While previously introduced methods for producing rankings, total orderings as well as partial orderings at chosen confidence-levels, can be directly applied to our data, determining which ranking is more accurate turns out to be methodologically and computationally more involved due to the specific nature of GEC outputs.

Agreement	Value	Degree
Inter-annotator	0.29	Weak
Intra-annotator	0.46	Moderate

(A) Inter-annotator and intra-annotator agreement for all judges

	1	2	3	4	5	6	7	8
1	.42	.26	.30	.37	.34	.26	.31	.24
2	–	.30	.25	.28	.23	.20	.10	.20
3	–	–	.50	.35	.44	.34	.46	.26
4	–	–	–	.34	.34	.30	.20	.26
5	–	–	–	–	.60	.36	.34	.32
6	–	–	–	–	–	.44	.35	.25
7	–	–	–	–	–	–	*	*
8	–	–	–	–	–	–	–	.48

(B) Pairwise inter-annotator and intra-annotator agreement per judge. Stars indicate too few overlapping judgements.

TABLE 4.3: Inter-annotator and intra-annotator agreement (Cohen's κ) on unexpanded pairwise judgments.

We adapted two ranking methods applied during WMT13 and WMT14 to human GEC evaluation: the Expected Wins method and a WMT-specific adaption of TrueSkill.

4.3.2.1 Ranking methods

The **Expected Wins** (EW) method has been introduced for WMT13 (Bojar et al., 2013) and is based on an underlying model of "relative ability" proposed in Koehn (2012). One advantage of this method is its intuitiveness — the scores reflect the probability that a system S_i is ranked higher than another system that has been randomly chosen from a pool of opponents $\{S_j : j \neq i\}$.

Defining the function $win(A, B)$ as the number of times system A is ranked better than system B, Bojar et al. (2013) calculate EW scores as follows:

$$\text{score}_{\text{EW}}(S_i) = \frac{1}{|\{S_j\}|} \sum_{j,j \neq i} \frac{\text{win}(S_i, S_j)}{\text{win}(S_i, S_j) + \text{win}(S_j, S_i)}.$$

This model ignores ties which we find slightly unsettling, as in our case the majority of judgments are in fact ties. Based on the findings of Bojar et al. (2011) that ties introduce bias, they are dropped from calculations of the rankings for the recent WMT human evaluation campaigns. Bojar et al. (2014) evaluated selected methods of computing rankings, including EW and the TrueSkill method discussed below with the conclusion to choose TrueSkill as the official ranking method.

The **TrueSkill** (TS) ranking system (Herbrich et al., 2007) is a skill based ranking system for Xbox Live developed at Microsoft Research. It is used to identify and model player's (GEC systems in our case) ability in a game to assign players to competitive matches. The TrueSkill ranking system models each player S_i by two parameters: the average relative ability μ_{S_i} and the degree of uncertainty in the player's ability $\sigma^2_{S_i}$.

Maintaining an uncertainty allows TS to make greater changes to the ability estimates at the beginning, smaller changes after a number of consistent matches has been played. Due to that TS can identify the ability of individual players from a smaller number of pairwise comparisons.

A modification of this approach to the WMT manual evaluation procedure by Sakaguchi et al. (2014) has been adopted as the official ranking method during WMT14 replacing EW. TrueSkill system scores are calculated as inferred means:

$$\text{score}_{TS}(S_i) = \mu_{S_i}.$$

The ranking is created by sorting systems accordingly to the obtained scores.

4.3.2.2 Rank clusters

Both ranking methods produce total orderings without information on statistic significance of the obtained ranks. In the context of MT, Bojar et al. (2014) notice that the similarity of the participants in terms of used methods and training data causes some of them to be very close in quality. Thus, the similar systems are grouped into equivalence classes based on a method proposed by Koehn (2012). Although applied methods and training data among the systems examined in this work are quite diverse, a great similarity of produced outputs is an inherent feature of the GEC task. The great number of output overlaps and pairwise ties confirm this.

Therefore, for each system S_j placed on rank r_j we also tried to determine true systems rank ranges $[r'_j, \ldots, r''_j]$ at a confidence-level of $p \leq 0.05$ and clusters of equivalent systems by following the procedure described by Koehn (2012). This was accomplished by applying bootstrap resampling: pairwise rankings were drawn from the set of judgments with multiple drawings. Based on this sample a new ranking was produced. After repeating this process a 1000 times the obtained 1000 ranks for S_j were sorted, and the top 25 and bottom 25 ranks were discarded. The interval of the remaining ranks served as the final rank range.

Next, these rank ranges were used to produce clusters of overlapping rank ranges. Final rankings for both methods are presented in Tables 4.4b and 4.4c, EW and TS, respectively. These rankings differ from the official CoNLL-2014 ranking (Tab. 4.4a) and will be discussed in Section 4.3.2.4.

4.3.2.3 Choosing the final ranking

The presented methods create quite similar, though distinct rankings. Only one of these rankings should be chosen as the final result of the human evaluation task.

We followed Bojar et al. (2014) who choose the method based on the ranking model's ability to predict pairwise rankings. Accuracy is computed by 100-fold cross-validation. For each fold a new ranking is trained from 99 parts, the left-over part serves as testing data. Accuracy is then averaged over all 100 test sets.

In the first step, we calculated the accuracy of the unclustered total orderings for non-ties only, discarding ties. As noticed before, a ranking based on model scores alone cannot predict ties. This requires equivalence classes. Bojar et al. (2014) define a draw radius r such that systems whose scores differ by less than that parameter are assigned to one cluster. The value of r is tuned to maximize accuracy for each fold via grid search.

In the case of GEC, due to the large number of ties, their method of tuning r is trapped in local maxima and assigns all systems to a single cluster. Alternatively, we propose to calculate

#	System	P	R	$M^2_{0.5}$
1	CAMB	39.7	30.1	37.3
2	CUUI	41.7	24.8	36.7
3	AMU	41.6	21.4	35.0
4	POST	34.5	21.7	30.8
5	NTHU	35.0	18.8	29.9
6	RAC	33.1	14.9	26.6
7	UMC	31.2	14.4	25.3
8	PKU	32.2	13.6	25.3
9	SJTU	30.1	05.1	15.1
10	UFC	70.0	01.7	07.8
11	IPN	11.2	02.8	07.1
12	IITB	30.7	01.3	05.9
13	INPUT	00.0	00.0	00.0

(A) Official CoNLL-2014 ranking

#	Score	Range	System
1	0.628	1	AMU
2	0.566	2-3	RAC
	0.561	2-4	CAMB
	0.550	3-5	CUUI
	0.539	4-5	POST
3	0.513	6-8	UFC
	0.506	6-8	PKU
	0.495	7-9	UMC
	0.485	7-10	IITB
	0.463	10-11	SJTU
	0.456	9-12	INPUT
	0.437	11-12	NTHU
4	0.300	13	IPN

(B) Human Expected Wins ranking

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

(C) Human TrueSkill ranking

TABLE 4.4: Comparison of official CoNLL-2014 ranking and human rankings. Ranges and clusters have been calculated with bootstrap resampling at $p \leq 0.05$.

clusters according to the method described in the previous section. This does not quite solve the problem of overfitting to ties, since decreasing the p-value has an effect similar to increasing r. However, by fixing $p \leq 0.05$ we directly evaluate rankings of the form given in Table 4.4. The absolute values of scores and their different interpretations between methods become irrelevant, which makes it unnecessary to tune a parameter like r.

The main drawback of this approach is its computational cost[10]. For each of the 100 folds we bootstraped another 100 rankings with EW and TS, fixed $p \leq 0.05$ and calculated rank clusters. The single clustered ranking for each fold was then used to calculate accuracy for the held-out test data.

Method	EW	TS
Total ordering (non-ties)	58.18	58.15
Bootstrapped clusters	40.12	39.48

TABLE 4.5: Accuracy for ranking-based prediction of pairwise judgments.

For our data, contrary to the MT-specific results from Bojar et al. (2014), EW beats TS in both cases (Table 4.5). Since the difference in quality for the total orderings are minimal, choosing a method based on the quality of the clustering for a common p seems reasonable. We therefore present the Expected Wins-based ranking (Table 4.4b) as the final result of the human evaluation effort and refer to it in the remainder of this chapter when the human ranking is mentioned.

[10]Computing 10000 TS rankings took about 12 hours using 32 parallel processes.

4.3.2.4 Analysis

The final human-created ranking (Table 4.4b) consists of four non-overlapping rank clusters. Rank ranges have been calculated at a confidence level of 95%. While general tendencies are similar, comparing the official CoNLL-2014 ranking (Table 4.4a) with the manually created Expected Wins ranking shows a few interesting differences:

- The AMU system, ranked third in the official automatic ranking, is judged to be the leader by human judges occupying its own rank cluster. For six out of eight judges, AMU has the highest score. Two judges place it second and rank ranges always start with 1. The score difference between AMU and the second-placed system corresponds to the difference between the second and seventh system.
- The officially winning system CAMB occupies third place in terms of EW scores and is placed in the second rank cluster with four other systems. Out of eight judges, only one put CAMB on first place in terms of EW scores.
- RAC, ranked in the middle of the official ranking, is elevated to second place although it occupies a rank cluster with three other systems.
- NTHU is a system that, based on M^2 scores, should be similar to RAC, is put on second to last position in terms of EW scores.
- Two systems are judged to be worse than the unchanged INPUT in terms of EW scores. This cannot be captured by the M^2 metric. The rank cluster of systems that includes INPUT is the largest among the four clusters and suggests these co-clustered system hardly differ from INPUT.

The revealed discrepancies put a question mark behind the idea of conducting competitions of GEC systems based on automatic metrics alone, at least as long the quality of the used metrics has not been established. Of course, the human evaluation performed in this work may be disputable as well. While we believe that the applied methods of computing rankings from pairwise judgments produce accurate system-level rankings under the given assumptions, we cannot help but wonder about the effects of a possible over-simplification.

Current ranking methods do not take into account annotator bias or quality. A closer look at the per-judge rankings provided in Table 4.6 seems to suggest that certain phenomena maybe due to the influence of single annotators. Including uncertainty about judgment quality into the ranking could help. This might be difficult on the level of pairwise judgments as per sentence inter-annotator agreement (Table 4.3) is naturally low for this kind of ambiguous task, but system-level inter-annotator ranking correlations in the next section (Table 4.9) are on average higher (though may still vary from 0.19 to 0.91) and may allow for a better assessment of annotator quality.

For completion, we also include pairwise comparisons between all systems according to EW in Table 4.7. Each cell contains the percentage of cases the system in that column was judged to be better than the system in that row. Bold values mark the winner. We applied the Sign Test to measure statistically significant differences: $\star$ indicates statistical significance at $p \leq 0.10$; † indicates statistical significance at $p \leq 0.05$; and ‡ indicates statistical significance at $p \leq 0.01$.

#	Score	Range	System
1	0.674	1-2	CAMB
	0.658	1-2	AMU
2	0.573	3-6	CUUI
	0.573	3-6	PKU
	0.566	3-7	POST
	0.553	3-8	RAC
	0.544	4-8	NTHU
	0.537	5-8	UMC
3	0.436	9-10	SJTU
	0.419	9-11	UFC
	0.387	10-12	IITB
	0.332	11-12	INPUT
4	0.276	13	IPN

(A) Judge 1

#	Score	Range	System
1	0.640	1	AMU
2	0.559	2-6	CUUI
	0.558	2-6	RAC
	0.557	2-7	UFC
	0.543	2-7	CAMB
	0.526	3-9	PKU
	0.511	5-10	POST
	0.511	4-11	IITB
	0.508	5-10	UMC
	0.473	7-11	INPUT
	0.472	8-11	NTHU
3	0.398	12	SJTU
4	0.286	13	IPN

(B) Judge 2

#	Score	Range	System
1	0.612	1-2	AMU
	0.578	2-3	CUUI
	0.564	2-5	UFC
	0.534	3-7	RAC
	0.526	4-7	POST
	0.517	4-8	UMC
	0.508	4-9	IITB
	0.474	5-11	INPUT
	0.467	8-12	SJTU
	0.464	8-12	PKU
	0.463	8-12	CAMB
	0.458	9-12	NTHU
2	0.333	13	IPN

(C) Judge 3

#	Score	Range	System
1	0.641	1-2	AMU
	0.631	1-2	CAMB
2	0.581	3-4	RAC
	0.578	3-4	CUUI
3	0.507	5-10	UFC
	0.494	5-10	POST
	0.490	5-10	UMC
	0.488	5-11	SJTU
	0.486	5-10	PKU
	0.473	5-11	INPUT
	0.441	9-12	NTHU
	0.378	11-12	IITB
4	0.313	13	IPN

(D) Judge 4

#	Score	Range	System
1	0.613	1-2	AMU
	0.608	1-2	RAC
2	0.568	3-5	CUUI
	0.554	3-6	CAMB
	0.535	4-7	POST
	0.526	4-9	UFC
	0.515	5-9	PKU
	0.496	6-10	SJTU
	0.487	6-11	INPUT
	0.472	8-11	IITB
	0.461	9-11	UMC
3	0.375	12	NTHU
4	0.290	13	IPN

(E) Judge 5

#	Score	Range	System
1	0.601	1-2	RAC
	0.579	1-4	AMU
	0.565	1-6	IITB
	0.562	2-6	POST
	0.548	2-8	INPUT
	0.535	3-8	UFC
	0.519	6-9	PKU
	0.516	6-9	CAMB
	0.491	7-10	SJTU
	0.474	9-10	CUUI
2	0.428	11	UMC
3	0.368	12	NTHU
4	0.313	13	IPN

(F) Judge 6

#	Score	Range	System
1	0.788	1	AMU
2	0.697	2	CAMB
3	0.553	3-7	RAC
	0.544	3-7	UMC
	0.537	3-7	POST
	0.533	3-10	IITB
	0.487	5-10	PKU
	0.474	5-12	INPUT
	0.462	6-11	CUUI
	0.436	6-12	SJTU
	0.408	8-12	UFC
	0.386	9-12	NTHU
4	0.303	13	IPN

(G) Judge 7

#	Score	Range	System
1	0.660	1-2	AMU
	0.625	1-3	CUUI
	0.568	2-7	IITB
	0.556	3-6	CAMB
	0.543	3-7	UMC
	0.537	3-7	POST
	0.483	5-10	INPUT
	0.481	5-10	UFC
	0.478	7-11	RAC
	0.466	7-11	NTHU
	0.420	10-12	PKU
	0.419	10-12	SJTU
2	0.260	13	IPN

(H) Judge 8

TABLE 4.6: Rankings by individual annotators. Cluster ranks and rank ranges have been computed with boostrap resampling at $p = 0.1$ to accomodate for the reduced number of judgments per ranking.

	AMU	RAC	CAMB	CUUI	POST	UFC	PKU	UMC	IITB	SJTU	INPUT	NTHU	IPN
AMU	–	.44 ‡	.47 ⋆	.46 †	.44 ‡	.34 ‡	.40 ‡	.37 ‡	.32 ‡	.34 ‡	.32 ‡	.31 ‡	.24 ‡
RAC	**.56** ‡	–	**.53**	.48	.48	.40 ‡	.45 †	.44 ‡	.39 ‡	.38 ‡	.38 ‡	.43 ‡	.28 ‡
CAMB	**.53** ⋆	.47	–	.49	.45 ‡	.43 ‡	.43 ‡	.42 ‡	.42 ‡	.43 ‡	.42 ‡	.43 ‡	.34 ‡
CUUI	**.54** †	**.52**	**.51**	–	.49	.42 ‡	.47	.46 †	.42 ‡	.41 ‡	.41 ‡	.42 ‡	.32 ‡
POST	**.56** ‡	**.52**	**.55** ‡	**.51**	–	.45 ‡	.47	.46 ⋆	.44 ‡	.44 ‡	.43 ‡	.42 ‡	.29 ‡
UFC	**.66** ‡	**.60** ‡	**.57** ‡	**.58** ‡	**.55** ‡	–	**.54** ⋆	**.50**	.49	.44 ⋆	.27 †	.42 ‡	.21 ‡
PKU	**.60** ‡	**.55** †	**.57** ‡	**.53**	**.53**	.46 ⋆	–	.50	.47	.46 ⋆	.46 ⋆	.46 †	.35 ‡
UMC	**.63** ‡	**.56** ‡	**.58** ‡	**.54** †	**.54** ⋆	.50	**.50**	–	.48	.47	.48	.45 ‡	.35 ‡
IITB	**.68** ‡	**.61** ‡	**.58** ‡	**.58** ‡	**.56** ‡	**.51**	**.53**	**.52**	–	.48	.43	.43 ‡	.27 ‡
SJTU	**.66** ‡	**.62** ‡	**.57** ‡	**.59** ‡	**.56** ‡	**.56** ⋆	**.54** ⋆	**.53**	**.52**	–	**.53**	.46 ⋆	.30 ‡
INPUT	**.68** ‡	**.62** ‡	**.58** ‡	**.59** ‡	**.57** ‡	**.73** †	**.54** ⋆	**.52**	**.57**	.47	–	.43 ‡	.22 ‡
NTHU	**.69** ‡	**.57** ‡	**.57** ‡	**.58** ‡	**.58** ‡	**.58** ‡	**.54** †	**.55** ‡	**.57** ‡	**.54** ⋆	**.57** ‡	–	.41 ‡
IPN	**.76** ‡	**.72** ‡	**.66** ‡	**.68** ‡	**.71** ‡	**.79** ‡	**.65** ‡	**.65** ‡	**.73** ‡	**.70** ‡	**.78** ‡	**.59** ‡	–

TABLE 4.7: Head-to-head comparison for human Expected Wins.

4.3.3 Correlation with GEC metrics

Since WMT08 (Callison-Burch et al., 2008) the "metrics task" has been part of the WMT. The aim of the metrics task is to assess the quality of automatic evaluation metrics for MT in terms of correlation with the collected human judgments. We attempt the same in the context of GEC.

4.3.3.1 Measures of correlation

Based on Macháček and Bojar (2013), we used Spearman's rank correlation ρ and Pearson's correlation co-efficient r to compare rankings produced by various metrics to the human-created ranking from the previous section. All metrics presented in Section 4.2.4 try to assign higher scores to better systems. Therefore metrics with Spearman's ρ or Pearson's r closer to 1 are deemed to be better. The comparison is made on the document-level, and we leave sentence-level correlation as a topic for future research.

The simplified version of **Spearman's rank correlation** ρ for rankings with no ties is defined as:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)},$$

where d_i is the difference between the human rank and the metric's rank for system i, n is the number of systems. If all system are ranked in the same order then $\rho = 1$, for the inverse order $\rho = -1$.

Macháček and Bojar (2013) find that Spearman's ρ is too harsh in cases where systems are very close in terms of EW scores but differ by rank and propose to also use **Pearson's correlation co-efficient** r. During the WMT14 metrics task (Macháček and Bojar, 2014b), Spearman's ρ has been replaced with Pearson's r entirely. The Pearson's correlation co-efficient r is calculated as:

Metric	Spearman's ρ	Pearson's r
M^2 $F_{1.0}$	0.648	0.610
M^2 $F_{0.5}$*	0.692	0.627
M^2 $F_{0.25}$	0.720	0.680
M^2 $F_{0.18}$	**0.758**	**0.701**
M^2 $F_{0.1}$	0.670	0.652
I-WAcc	-0.154	-0.098
BLEU	-0.346	-0.240
METEOR	-0.374	-0.241

TABLE 4.8: Correlation results for various metrics and human ranking. Starred metric has been used for the CoNLL-2014 shared task ranking.

$$r = \frac{\sum_{i=1}^{n}(H_i - \bar{H})(M_i - \bar{M})}{\sqrt{\sum_{i=1}^{n}(H_i - \bar{H})^2}\sqrt{\sum_{i=1}^{n}(M_i - \bar{M})^2}},$$

where H is the vector of human scores, M is the vector of metric scores, $\bar{H}$ and $\bar{M}$ are corresponding means.

4.3.3.2 Metrics

The inventory of automatic evaluation metrics for GEC is much more restricted than in the case of MT. We assessed four metrics introduced earlier in Section 4.2.4: MaxMatch, I-WAcc, BLEU and METEOR.

The default parameters for English in METEOR metric have been selected to match human-created MT-system rankings from the earlier editions of WMT. Tuning METEOR parameters for GEC-specific results was out of the scope of this thesis.

In order to be able to use the CoNLL-2014 gold standard with MT metrics, the edit-based annotation has been converted into two plain text files, one per annotator, that are treated as target language references.

4.3.3.3 Analysis

Correlation results are collected in Table 4.8. Assumed degrees of correlation are $[0, 0.3]$ for negligible correlation, $[0.3, 0.5]$ for low correlation, $[0.5, 0.7]$ for moderate correlation, and $[0.7, 0.9]$ and $[0.9, 1]$ for high and very high correlations (Hinkle et al., 2003). The M^2 metric is generally moderately or strongly correlated with human judgment, on the brink of high correlation for values of β closer to 0.2. Compared to M^2, the other metrics are weakly inversely correlated to human judgment. Inverse correlation with human judgments for metrics that all assign higher scores to better systems seems problematic. In the case of I-WAcc, we would go as far as stating a near-absence of correlation. It seems the very conservative approach adopted for I-WAcc does not correspond to the notion of quality that our judges worked out for themselves.

As can be seen, the switch to $\beta = 0.5$ from $\beta = 1.0$ for the CoNLL-2014 shared task was a good choice. A higher correlation can be achieved by weighting precision even higher with $\beta = 0.25$ and the maximum is reached for $\beta = 0.18$. However, correlation drops sharply for

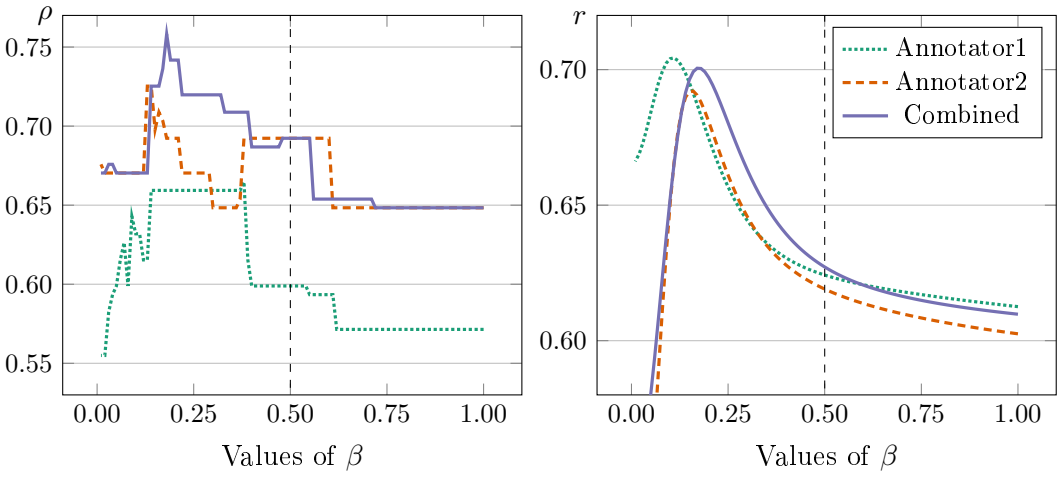


FIGURE 4.4: Spearman's ρ and Pearson's r correlation of M^2 with human judgment w.r.t. β. Dashed line marks official CoNLL-2014 choice $\beta = 0.5$.

	1	2	3	4	5	6	7	8	ρ	$\bar{\rho}$
1	–	.70	.31	.76	.74	.19	.62	.48	.70	
2	.72	–	.77	.84	.90	.57	.59	.64	.93	
3	.53	.89	–	.66	.70	.58	.42	.64	.63	
4	.82	.79	.69	–	.91	.42	.67	.54	.91	.72
5	.65	.85	.82	.87	–	.63	.63	.51	.93	
6	.32	.71	.67	.56	.86	–	.63	.39	.42	
7	.72	.74	.57	.76	.72	.63	–	.63	.76	
8	.64	.85	.86	.69	.72	.57	.75	–	.60	
r	.67	.93	.82	.87	.92	.66	.80	.82	–	
$\bar{r}$					.80					

TABLE 4.9: Inter-annotator correlation (Spearman's ρ above the diagonal, Pearson's r below) between rankings computed for individual judges.

$\beta = 0.1$ which adds risk to choosing a good value. The lack of positive correlation for the MT-metrics is surprising in the light of improvement that results from a shift towards precision for M^2, as for instance BLEU is based on precision.

Figure 4.4 contains detailed plots of ρ and r with regard to β within the $[0, 1]$ range. The values of ρ and r hardly change for $\beta > 1$. As the CoNLL-2014 test data included annotation from two annotators, we plot correlation w.r.t β for both annotators separately and for the official combined gold standard. In the case of Spearman's ρ alternative error annotations leads to higher correlation values. Based on the plots we would recommend a settings of $0.2 \leq \beta \leq 0.3$. It is of course unclear whether this conclusion can be generalized across other shared tasks or languages.

Inter-annotator correlations of rankings computed for individual judges (Table 4.9) can be treated as human-level upper bounds for metric correlation. The penultimate column and row contain correlations of rankings for individual judges with rankings computed from all judges minus the respective judge. The last column and row contain the respective weighted (w.r.t. judgments per judge) average of these correlations. These averages can be interpreted as approximate correlations of rankings produced by new human judges with the current ranking. With values

between 0.60 and 0.93 (with one outlier at 0.42) it seems that the performance of M^2 is close to the range of human capabilities.

4.4 Summary

The choice of automatic evaluation metric is an important decision, which influences the progress in the field. All foregoing shared tasks in automated grammatical error correction use automatic metrics to determine the quality of the participating systems. We have successfully adapted methods from the WMT human evaluation campaigns to GEC and shown that the commonly used parameters for standard metrics in these shared tasks may not be optimal. Nevertheless, the MaxMatch metric turned out to be best correlated with human judgments among tested metrics. The maximum is reached for $\beta = 0.18$. This justifies empirically the shift from $F_{1.0}$ to $F_{0.5}$ between CoNLL-2013 and CoNLL-2014 shared tasks.

In experiments presented in the remainder of this thesis, we use the M^2 metric with $F_{0.5}$-score as our main evaluation metric. This allows for an meaningful comparison to the systems participating in the shared task and, more importantly, to the state-of-the-art results reported on the CoNLL-2014 test set in follow-up research.

Chapter 5

Grammatical Error Correction Using Statistical Machine Translation

The application of statistical machine translation to grammatical error correction has been reported as early as 2006 (Brockett et al.). Recently, it has become a popular approach that is used in systems achieving state-of-the-art results. In this chapter we present our contribution to GEC by machine translation using the Moses toolkit (Koehn et al., 2007)[1]. Two important ideas are introduced: parameter tuning towards the task-specific evaluation metric and the exploration of different dense feature functions. We find that a bare-bones phrase-based SMT setup with proper optimization outperforms all previously published results for the CoNLL-2014 test set by a large margin while being trained on the same, publicly available data.

The chapter is organized as follows. Section 5.1 provides a background on the phrase-based statistical machine translation. In Section 5.2, we introduce task-specific feature functions used in experiments. Training and test sets are described in Section 5.3. The experiments on tuning and optimization of SMT-based GEC models, using additional task-specific features, and utilizing additional parallel and monolingual data are presented in Section 5.4. We summarize in Section 5.5.

5.1 Statistical machine translation

The objective of machine translation is to translate an input sentence S in one language into an output sentence $\hat{T}$ in another language, which is the most probable translation of S:

$$\hat{T} = \arg\max_{T} p(T|S).$$

The process of finding the most probable translation is known as *decoding*.

In phrase-based statistical machine translation (Koehn et al., 2003), the decoder translates phrases (strings of words) then concatenates the translations together into a sentence. Hypotheses are generated from the *phrase translation table* (or *phrase table*), which consists of potential translations of phrases and their probabilities (*translation options*). The probabilities are learnt from parallel training data using word-to-word alignment between source and target sentence pairs.

[1] The chapter presents an extended work originally published in Junczys-Dowmunt and Grundkiewicz (2014) and Junczys-Dowmunt and Grundkiewicz (2016).

5.1.1 Log-linear model

The currently dominant approach in phrase-based machine translation is the use of *log-linear* combinations of *feature functions* to model the probability $p(T|S)$:

$$p(T|S) = \prod_{i=1}^{N} h_i(T|S)^{\lambda_i},$$
$$\log p(T|S) = \sum_{i=1}^{N} \lambda_i \log h_i(T|S),$$

where h_i is a feature function with the corresponding feature weight λ_i, and N is the total number of feature functions. Typically, two types of features are distinguished: stateless features that make independence assumption at the boundaries between phrases, and stateful features that cross phrase boundaries. One may define features of varying complexity that depend on any properties of the source and hypothesis sentences.

In Moses (Koehn et al., 2007), standard *stateless features* defined for each phrase pair are: probability of the target phrase given the source phrase, probability of the source phrase given the target phrase, word-level probability of the target phrase given words in the source phrase, and vice versa. Other common features include penalties on the number of phrases used, the number of words on the target side of a phrase, and the number of unknown words.

Stateful features, in contrast to stateless features, depend not only on the current phrase translation, but also on prior translation decisions using Markov chain rule. A common stateful feature is the probabilistic n-gram language model, which estimates the probability of the target sentence independently of the source sentence.

5.1.2 Model training and tuning

Model parameters, i.e feature values and feature weights, are estimated from training data. Most feature parameters are learnt from a corpus of parallel text, although language models are estimated on the target-side language only.

The feature weights λ of the log-linear model are tuned on a development set usually smaller than the training corpus with regard to the chosen evaluation method. The most commonly used optimization algorithm for this purpose is *Minimum Error Rate Training* (MERT) (Och, 2003).

Moses comes with tools that can tune parameter vectors according to different MT tuning metrics. The default combination is MERT and BLEU (Papineni et al., 2002). Skipping parameter vector tuning can be perceived as methodologically incorrect as likely suboptimal results from the untuned systems make the comparison of various system configurations untrustworthy.

Researchers applying SMT to grammatical error correction often ignore the proper tuning procedure in their systems, which is incorrect. The exceptions are Brockett et al. (2006) and Susanto et al. (2014), who mention minimum error rate training according to BLEU. Dahlmeier and Ng (2012a) build a custom beam-search decoder for GEC and perform parameter tuning with $M^2_{1.0}$. The authors find PRO (Hopkins and May, 2011) to work better with M^2 than MERT. Their specialized decoder is compared to Moses that has been tuned with BLEU, which may not be a fair comparison (as we will show in Section 5.4.1). None of the CoNLL-2013 SMT-based

Source phrase	Target phrase	d_{LD}	d_D	d_I	d_S
a short time .	short term only .	3	1	1	1
a situation	into a situation	1	0	1	0
a supermarket .	a supermarket .	0	0	0	0
a supermarket .	at a supermarket	2	1	1	0
able	unable	1	0	0	1

TABLE 5.1: Word-based Levenshtein distances and separated edit operations.

systems seems to use parameter tuning, even the winning system of Felice et al. (2014). For Chinese error correction, Zhao and Ishikawa (2015) built a hierarchical SMT system tuned with a linear combination of evaluation metrics using MERT. None of the papers on QALB shared task for Arabic (Mohit et al., 2014a, Rozovskaya et al., 2015) mentions a task-specific tuning procedure for SMT-based systems, which shows how commonly SMT frameworks have been used in a black box manner for GEC-related tasks.

5.2 Feature functions

Feature engineering is essential in many machine learning applications. The standard features in SMT are geared towards modeling and guiding the translation process.

In the GEC setting, the most natural units on which features can be defined seem to be minimum edit operations that can be either counted or modelled with varying degrees of generalization. That way, the decoder can be informed on several levels of abstraction how the output differs from the input. In this section we introduce several stateless and stateful features that try to capture edits in isolation or in context.

All features described in this chapter are *dense features*, i.e. features that contribute to the overall score of each hypothesis.

5.2.1 Stateless features

Our stateless features are computed during translation option generation before decoding, modeling relations between source and target phrases. They are meant to extend the standard SMT-specific MLE-based phrase and word translation probabilities with meaningful phrase-level information about the correction process.

Levenshtein distance: The edit distance between source and target phrases can be used as a translation model feature. We use the word-based Levenshtein Distance (LD) (Levenshtein, 1966) that computes the minimum number of operations such as word insertion, deletion or substitution, required to convert the source phrase into the target phrase. The count informs the model how many transformations of a source phrase are needed for the correction to be made. Examples of Levenshtein distance for pairs of phrases are presented in Table 5.1.

The Levenshtein distance can be also calculated on character-level as in Felice et al. (2014).

Edit operation counts: Edit operation counts are more informative than the Levenshtein distance. We separately count numbers of deletions (D), insertions (I), and substitutions (S) that transform the source phrase into the target phrase. Table 5.1 provides examples.

Each edit operation count contribute to the log-linear model as a separate feature function, which allows to weight different edit operation types separately:

$$\log p(T|S) = \sum_{i=1}^{N} \lambda_i \log h_i(T|S) + \lambda_D h_D(T|S) + \lambda_I h_I(T|S) + \lambda_S h_S(T|S).$$

To maintain additivity in the log-linear model, instead of using raw distance counts i, we use $\exp(i)$ as scores in the translation model:

$$\begin{aligned} h_{LD} &= \exp(d_{LD}), \\ h_D &= \exp(d_D), \\ h_I &= \exp(d_I), \\ h_S &= \exp(d_S). \end{aligned}$$

For both feature types, the total sum of counts for a sentence is roughly equal to the Levenshtein distance between the source and target sentence[2].

5.2.2 Stateful features

Contrary to stateless features, stateful features can "look" at translation hypotheses outside their own span and take advantage of the constructed target context. We experiment with several LM-based features computed on various levels of generalization of the target: tokens, part of speech tags, and automatic word classes. We also introduce edit-based stateful features in the form of bilingual models.

N-gram language model: Language models are the most typical stateful feature in statistical machine translation (Koehn et al., 2007) controlling the fluency of the output sentence. They estimate how probable a text is with respect to a given corpus. The n-gram language model calculates the probability of a target sentence T based on the previous $n-1$ word context using higher-order Markov models (Markov, 1960):

$$p_{\text{LM}}(T) \approx \prod_{i=1}^{|T|} p(t_i|t_{i-n+1}, \ldots, t_{i-1}).$$

For our experiments, we built 5-gram language models with modified Kneser-Ney smoothing (Chen and Goodman, 1996) using KenLM (Heafield et al., 2013).

Part-of-speech n-gram language model: Language models can be estimated on target-side factors such as morphosyntactic Part-of-Speech (POS) tags. Since the total number of distinct POS tags is significantly smaller compared to the number of words, a Part-Of-Speech Language Model (POS LM) can be estimated for higher-order n-grams than for token-level LMs. Hence, they are better at capturing long-distance dependencies in a sentence.

[2] The sum may be larger than the minimum edit distance between source and target sentence as edits cannot be computed across phrase boundaries.

Cluster id	Most frequent words
1	is,was,has,had,about,became,did,began,said,does,wrote,means
36	of,for,with,on,from,at,after,into,during,between,over,while
37	even,much,very,long,great,good,little,personal,strong,too,real
42	government,political,war,Soviet,Christian,religious,party
45	London,England,Hall,Scotland,Wales,Victoria,Manchester,Kent,Town,Dublin
48	used,known,called,found,published,given,considered,developed
123	1,2,3,4,5,6,8,7,9,0,64,63,62,98,61,76,79,OH,93,89,74,105,101,450,WK
130	then,thus,therefore,hence,thereby,alternatively,presumably,nowadays

TABLE 5.2: Clusters of automatic word-classes.

We run the Stanford Log-linear Part-of-Speech Tagger[3] (Toutanova et al., 2003) on Wikipedia data to obtain part-of-speech sequences, then we trained a 9-gram language model with KenLM. The tagset consists of 43 tags.

Word-class language model: Word-Class Language Models (WCLMs) are language models estimated on automatically extracted word clusters (Brown et al., 1992). WCLM is meant to work similarly to a LM trained on morphosyntactic POS tags. It is built on automatic word clusters, which are extracted from plain texts using language-independent unsupervised machine learning methods. Also, the number of word clusters can be adjusted to a specific task if needed.

Our word clusters are calculated with `word2vec`[4] (Mikolov et al., 2013) based on the Wikipedia data used for language model training. We built 200-dimensional vectors of word embeddings using default settings and grouped them into 200 clusters. Example clusters are presented in Table 5.2.

We computed a 9-gram word-class language model from the Wikipedia data with all tokens replaced with corresponding word cluster identifiers.

Operation sequence model: Durrani et al. (2013, 2011) introduce the Operation Sequence Model (OSM) for phrase-based statistical machine translation in Moses. These models are Markov translation models estimated on a linear sequence of operations $(o_1, o_2, \ldots, o_J)$ generated from the source sentence S, the target sentence T and their word-alignment:

$$p_{\text{OSM}}(T, S) \approx \prod_{j=1}^{J} p(o_j | o_{j-n+1}, \ldots, o_{j-1}).$$

Operations define steps that are needed to generate the sequence T from S. Typical operations are: generate the target phrase t from the source phrase s, generate the source/target only, generate identical word(s), insert a gap, jump forward or backward, etc. An example of a generated operation sequence for token-based OSM is presented in Figure 5.3.

OSM integrates reordering operations and is perceived as a context-aware reordering model. In our setting, we explicitly forbid reordering operations, which reduces the OSM to a context-aware edition model. Translations between identical words are matches, translations that have different words on source and target sides are substitutions. Insertions and deletions work in

[3] http://nlp.stanford.edu/software/tagger.shtml
[4] https://code.google.com/p/word2vec/

Source	nowadays , people are more health conscious .
Target	nowadays , people have been more health conscious than before .
Operations	_TRANS_nowadays_TO_nowadays _TRANS_,_TO_, _TRANS_people_TO_people _TRANS_are_TO_have _INS_been _TRANS_more_TO_more _TRANS_health_TO_health _TRANS_conscious_TO_conscious _INS_than _INS_before _TRANS_._TO_.

TABLE 5.3: Example of the operation sequence for OSM.

the same way as in original OSM. Gaps, jumps, and other operations typical for OSMs that represent reordering do not appear.

We trained a 5-gram language model on operations generated from the source and target sentences from the training data.

Bilingual neural language model: Neural network joint models (NNJMs) have been introduced as a feature function in Moses by Devlin et al. (2014). These are bilingual language models which augment feedforward neural language models (Bengio et al., 2003) with a source context window. For a source sentence S and a target sentence T, the n-gram neural network joint model with m-word source window defines a conditional probability distribution as:

$$P_{\text{NNJM}}(T|S) \approx \prod_{i=1}^{|T|} p(t_i|t_i, \ldots, t_{i-n+1}, \mathcal{S}_i^m),$$

where $\mathcal{S}_i^m$ is an m-word source window for a target word t_i based on the word alignment between sentences S and T.

We used the Neural Probabilistic Language Model Toolkit[5] (Vaswani et al., 2013) to build a 5-gram language model with 9 source context words.

5.3 Training and test data

In this section we introduce data sets that we will use in further experiments. Next, we discuss the differences between data sets in respect to measures based on edit frequencies.

5.3.1 Parallel data

NUCLE: We use the NUS Corpus of English in version 3.2[6]. It serves as a training set and/or development set. We evaluate our methods on the CoNLL-2013, CoNLL-2014 and GEC-10 test sets. NUCLE data sets are summarized in Table 5.4.

Lang-8: We collecte all entries from "Lang-8 Learner Corpora v1.0"[7] with English as the learned language, we do not care about the native language of the user. Only entries for which at least one sentence has been corrected are taken into account. Sentences without corrections from such entries are treated as error-free and mirrored on the target side of the corpus. We

[5] http://nlg.isi.edu/software/nplm/

[6] http://www.comp.nus.edu.sg/~nlp/conll14st.html#nucle32

[7] The corpus has been released in December, 2012.

Corpus	Sentences	Tokens
NUCLE	57,151	1,161,567
CoNLL-2013 Test Set	1,381	29,207
CoNLL-2014 Test Set	1,312	30,144
GEC-10 Test Set	1,312	30,144
Lang-8 NAIST	2,186,460	25,732,858
Lang-8 WEB	3,475,848	48,398,304
WikEd	24,392,765	554,926,293

TABLE 5.4: Statistics of parallel training data.

do not remove alternative versions of the correction for a single sentence. Eventually, we obtain a corpus of 2,186,460 sentence pairs with 25,732,858 tokens on the uncorrected source side (Table 5.4). We call this resource Lang-8 NAIST.

We create a larger version of the corpus by crawling Lang-8 for additional entries[8]. We manage to nearly double the size of the corpus to 3,475,848 sentences with 48,398,304 tokens on the source side. In Table 5.4, this joint resource is labeled as Lang-8 WEB.

WikEd: We use the WikEd Error Corpus as additional training data. Only sentences not marked as potentially harmful are taken into account. After tokenization and true-casing of the data, the resource consists of 24,392,765 sentences and 554,926,293 tokens on the source side.

5.3.2 Monolingual data

Wikipedia: We extracted all text from the English version of Wikipedia[9]. A total number of ca. 213.08 million sentences is collected.

Common Crawl: We use the Common Crawl data made-available by Buck et al. (2014) and select it with cross-entropy filtering (Moore and Lewis, 2010) using the corrected NUCLE corpus as seed data. This procedure is a domain adaptation technique, which aims at selecting sentences the most similar to NUCLE data, and, as a side effect, decreases the size of the Common Crawl data. We keep all sentence with a negative cross-entropy score, which resulted in roughly 300GB of compressed text consists of ca. 59.13 billion of sentences. Statistics are presented in Table 5.5.

Corpus	Sentences	Tokens
Wikipedia	213.08 M	3.37 G
Common Crawl	59.13 G	975.63 G

TABLE 5.5: Statistics of monolingual training data.

5.3.3 Error rates

Learner error corpora differ from one another in several aspects: essays are written by learners with different native language (L1) and language proficiency; texts cover various topics and

[8] Additional data were scrapped in March, 2014.
[9] Wikipedia database dump from December 2nd, 2013: http://dumps.wikimedia.org/enwiki/20131202/

Corpus	Sub.	Del.	Ins.	WER	SER
NUCLE	0.5646	0.2221	0.2134	0.0609	0.3760
CoNLL-2013 Test Set	0.6304	0.2172	0.1525	0.1476	0.8088
CoNLL-2014 Test Set	0.6341	0.1860	0.1799	0.1137	0.7466
GEC-10 Test Set	0.6352	0.1766	0.1882	0.1427	0.8453
Lang-8 NAIST	0.5565	0.1370	0.3064	0.2227	0.5827
Lang-8 WEB	0.5690	0.1421	0.2890	0.2095	0.8356
WikEd	0.6089	0.1645	0.2266	0.0819	1.0000

TABLE 5.6: Comparison of error rates in parallel corpora.

style; the annotation schemas among scientific centres building error corpora may vary, or even annotators' preferences may have impact on the annotations of the same data sets (Bryant and Ng, 2015).

Word Error Rate (WER) is a common metric for the performance of speech recognition or machine translation systems. Even if it is not used as an evaluation metric in GEC, it can be used to characterize error corpora. WER is defined as:

$$\text{WER} = \frac{S + D + I}{N},$$

where S is the number of words substituted by a corrector, D is the number of deleted words, and I is the number of inserted words. N is the total number of words in the corrected sentence.

Sentence Error Rate (SER) is defined as the fraction of sentences that contain one or more edits, i.e.:

$$\text{SER} = \frac{|\text{sentences with} \geq 1 \text{ edits}|}{M}.$$

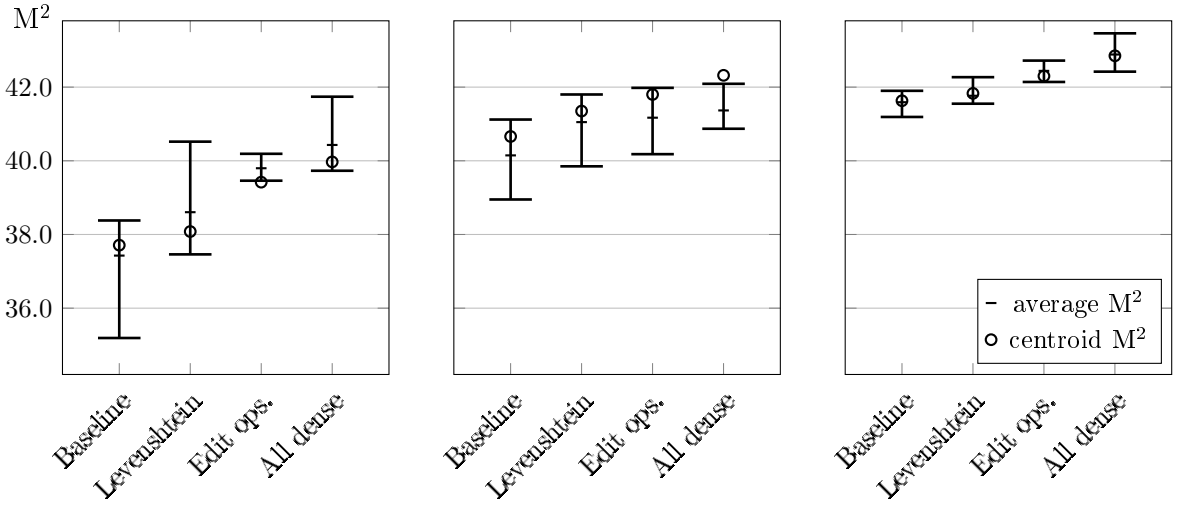
M is the total number of sentences in the data set.

Error rates, including distribution of substitutions, deletions and insertions, are presented in Table 5.6. Edits are extracted with LCS algorithm (Maier, 1978). Results are calculated on tokenized and true-cased sentences. Statistics for data sets annotated by two or more annotators have been averaged.

The NUCLE corpus contains less edits than test sets from both CoNLL shared tasks, which results in lower WER and SER. The Lang-8 corpus has a higher error frequency than NUCLE. High SER in WikEd comes from the fact that we extracted only sentences containing one or more edits. However, the lower WER in respect to high SER indicates shorter edits (in terms of the number of words being a part of an edit) which can be more typical for errors made by native-speakers. More than half of the edits in each corpus are substitutions, but the Lang-8 corpora contain more insertions. We observed that some corrected sentences in Lang-8 include comments from annotators.

5.4 Experiments

We run our experiments using the phrase-based SMT system Moses (Koehn et al., 2007). Only plain text data are used for translation model and language model training. When parameter



(A) Optimized using BLEU on the CoNLL-2013 test set

(B) Optimized using M^2 on the CoNLL-2013 test set

(C) Optimized using M^2 on 4 folds of error-rate-adapted NUCLE

FIGURE 5.1: Results on the CoNLL-2014 test set for different optimization settings and feature sets.

tuning is performed with the M^2 metric, the tuning set is provided in the M^2 format, which contains error annotations that introduce external linguistic knowledge.

The translation model is built with standard Moses training script. Word-alignment models are produced with MGIZA++ (Gao and Vogel, 2008). We restrict the word alignment training to 5 iterations of Model 1 and 5 iterations of the HMM-Model as we saw no differences in quality between this faster non-standard training sequence and the full MGIZA++ sequence with iterations of Model 3 and 4. Phrase tables are binarized with the compact phrase table (Junczys-Dowmunt, 2012). We do not use reordering models — the distortion limit is set to 0, effectively prohibiting any reordering.

All systems use one 5-gram language model that has been estimated from the target side of the parallel data available for translation model training and another 5-gram language model based on Wikipedia. We combine multiple language models as features in the log-linear model of Moses.

5.4.1 Tuning and optimization

In Figure 5.1 we present the results of three optimization scenarios for four feature function sets each. *Baseline* systems have no additional task-specific features (i.e. Levenshtein distance, edit operation counts, WCLM, POSLM, OSM, NNJM), which corresponds to a vanilla Moses setup. *Levenshtein* systems use the word-based Levenshtein distance feature. Levenshtein distance has been replaced by edit operation counts in systems described as *Edit ops.* *All dense* features include OSM, WCLM, and edit operation features. Systems are retuned when new features of any type are added.

Based on Clark et al. (2011) concerning the effects of optimizer instability, we report results averaged over five tuning runs. For each experiment in Figure 5.1, we outline the range between

the lowest and the highest M^2 scores from individual tunings. The small horizontal bars indicate the averaged M^2 score.

Additionally, we compute parameter weight vector centroids as suggested by Cettolo et al. (2011). They showed that parameter vector centroids averaged over several tuning runs yield similar to or better than average results and reduce variance. We generally confirm this for M^2-based tuning. In Figure 5.1 results for averaged weights vectors are marked as small circles. These are chosen as the final results.

5.4.1.1 Tuning towards BLEU

As mentioned earlier, the default combination of optimization algorithm and evaluation metric used to tune model parameters in Moses is MERT (Och, 2003) and BLEU (Papineni et al., 2002). It might be tempting to use these out-of-the-box settings also for GEC, but this is not an optimal solution if maximizing M^2 score is the final goal.

Junczys-Dowmunt and Grundkiewicz (2014) have already shown that tuning with BLEU is counterproductive in settings with M^2 as the evaluation metric. For inherently weak systems this can result in all correction attempts being disabled. This is due to MERT, which learns to disallow all changes since they lower the similarity to the reference as determined by BLEU. Systems with better training data, can be tuned with BLEU without suffering this negative effect. Despite these observations, Susanto et al. (2014) choose to tune the feature function weights of their two SMT-based systems with BLEU and report state-of-the-art results.

We first successfully reproduce results of Susanto et al. (2014) for BLEU-based tuning on the CoNLL-2013 test set as the development set (Fig. 5.1a), but got better results. We use similar training data, i.e. NUCLE and Lang-8 NAIST corpora. Susanto et al. (2014) report scores for their SMT-ML combinations of 37.90–39.39%. Repeated tuning runs place these scores within the range of possible values for a purely Moses-based system without any specific features (35.19–38.38%, Fig. 5.1a, *baseline* system) or with just the Levenshtein distance features (37.46–40.52%, Fig 5.1b, *Levenshtein* system). Since the authors do not report results for multiple tuning steps, the extend of influence of optimizer instability on their experiments remains unclear.

Even with BLEU-based tuning, we can see significant improvements when replacing Levenshtein distance with the finer-grained edit operations, and another performance jump with additional stateful features.

5.4.1.2 Tuning towards M^2

We extended MERT with the capability of tuning with respect to M^2. We re-implemented the M^2 metric in C++ and added it as a scorer to the Moses parameter optimization framework[10].

Tuning directly with M^2 (Fig. 5.1b) and averaging weights across five iterations, yields between 40.66% M^2 for a vanilla Moses system and 42.32% for a system with best dense features. Using the same training data and tuning data as the top-results reported for the CoNLL-2014 shared task to-date (39.39%), vanilla Moses outperforms these systems due to M^2-tuning alone. Task-specific features further increase that performance advantage.

Results of M^2-tuned systems seem to be more stable than those by systems tuned towards BLEU. Averaging weight vectors across runs to produce the final vector seems to be a good

[10] The C++ version of M^2 scorer is available from the main Moses repository: https://github.com/moses-smt/mosesdecoder

Algorithm 1 Training procedure using k-fold cross validation.

Input: Development set D (NUCLE), train set T (Lang-8 NAIST), the number of folds k (4), the number of MERT runs n (5)

Divide D into approximately equally sized k parts D_i
for each fold $i \in \{1 \dots k\}$ **do**
    $D_i^{train} \leftarrow \{D_j : j \neq i\}$
    $D_i^{tune} \leftarrow D_i$ with adapted error rate
    $T_i \leftarrow T + D_i^{train}$
    Train model M_i using training data T_i
    for $j = 1 \dots n$ **do**
        $w_{i,j} \leftarrow$ tune parameter weights in model M_i on D_i^{tune}
    end for
end for
$T_{final} \leftarrow T + D$
Train final model M_{final} using training data T_{final}
for $j = 1 \dots n$ **do**
    $w_j^{avg} \leftarrow$ average parameter weights $w_{i,j}$ for $i \in \{1 \dots k\}$
    Evaluate model M_{final} using weights w_j^{avg}
end for
$w^{avg} \leftarrow$ average parameter weights w_j^{avg} for $j \in \{1 \dots n\}$
Evaluate model M_{final} using weights w^{avg}

strategy: performance with the averaged weight vectors is either similar to or better than the average number for five runs.

5.4.1.3 Choosing development sets

No less important than choosing the correct tuning metric is a good choice of the development set. Among MT researches, there is a number of more or less well known truths about suitable development sets for translation-focused settings: usually they consist of between 2000 and 3000 sentences, which should be a good representation of the testing data. The official CoNLL-2013 test set consists of 1380 sentences, which might be barely enough for a translation-task, and it is unclear how to quantify it in the context of grammar correction. The low word error rate reveals that only 14.76% of the tokens are part of an erroneous fragment. This seems to be very little significant data for tuning an SMT system. We decided to take advantage of a larger development set in the form of the entire NUCLE corpus, which consists of 57,151 sentences. So far, NUCLE has only been used as translation model training data.

The word error rate of NUCLE is significantly lower than in the CoNLL-2013 test set (6.09% vs. 14.76%). We increase the error rate by greedily removing sentences from NUCLE until an error rate of ca. 15% is reached. This procedure causes 23,381 of 57,151 sentences and most error annotations to remain.

We further use a 4-fold cross validation: the NUCLE data is divided into four folds, and each fold serves as a development set for parameter tuning, while the three remaining parts are treated as translation model training data. The full Lang-8 NAIST corpus, which constitutes the remaining training data as in the previous experiments, is concatenated with the NUCLE training set, and four models are trained. Tuning is performed four times and the resulting four parameter weight vectors are averaged into a single weight vector across folds. We repeat

System	CoNLL-2013 P	CoNLL-2013 R	CoNLL-2013 M^2	CoNLL-2014 P	CoNLL-2014 R	CoNLL-2014 M^2
Baseline	43.00	16.06	32.20	48.97	26.03	41.63
+Levenshtein	44.09	16.50	33.04	49.79	25.51	41.83
+Edit ops.	44.41	16.99	33.57	49.84	26.35	42.30
+Levenshtein	45.63	16.90	**34.06**	51.02	26.17	**42.88**
+Edit ops. (char)	41.76	15.37	31.09	48.18	25.91	41.11
Baseline+Edit ops.	44.41	16.99	**33.57**	49.84	26.35	42.30
+POSLM	44.17	15.00	31.80	51.01	25.52	42.52
+WCLM	43.88	15.86	32.43	51.01	25.75	**42.64**
+POSLM	44.64	14.91	31.91	51.38	24.60	42.19
Baseline+Edit ops.	44.41	16.99	33.57	49.84	26.35	42.30
+NNJM	44.04	16.44	32.97	50.76	26.60	42.96
+OSM	46.47	17.31	**34.76**	50.73	26.27	42.77
+NNJM	45.07	17.16	34.01	50.55	27.16	**43.12**
Baseline+Edit ops.+WCLM+OSM	45.07	16.53	33.50	50.94	26.21	42.85

TABLE 5.7: The SMT system performance with various dense feature functions.

this procedure again five times, which results in 20 separate tuning steps. Final results on the CoNLL-2014 test set are obtained using the full translation model with a parameter vector average across five runs. The training procedure using k-fold cross validation is described in Algorithm 1.

The CoNLL-2013 test set is not being used for tuning and can serve as a second test set. As can be seen in Figure 5.1c, this procedure significantly improves performance, also for the *baseline* set-up (41.63%). The lower variance between iterations is an effect of averaging across folds.

5.4.2 Experiments with additional features

We examined the usefulness of particular feature functions for the SMT-based GEC system and summarized the results in Table 5.7. The *baseline* system uses two previously described language models and no edit operation features.

Firstly, we inspect the combination of various edit-based features added to the translation model (top part of Table 5.7). Adding word-based Levenshtein distance score to the baseline system trained on NUCLE and Lang-8 data improves the system performance by +0.84%. Even higher improvement is achieved with edit operation counts (+1.37%), or by combining both features as four scores (+1.86%). However, the character-level edit operation counts decrease the system output quality according to the two test sets.

Next, we examine POSLM and WCLM features (middle part of Table 5.7). Both are estimated on processed the Wikipedia texts. The *baseline* system already uses the token-based language model trained on Wikipedia. Adding a LM trained on part-of-speech tags or automatic word classes has only small effect on the system performance. Neither of the two increases the F-score on CoNLL-2013 test set, whereas the improvement on CoNLL-2014 test set is not significant: +0.22% and +0.34% for POSLM and WCLM respectively.

System	CoNLL-2013			CoNLL-2014		
	P	R	M^2	P	R	M^2
Baseline	36.07	11.30	25.07	42.93	18.64	34.05
+WikiLM	43.00	16.06	32.20	48.97	26.03	41.63
+CCLM	47.58	14.50	32.68	58.91	25.05	46.37
Dense	38.03	14.27	28.53	42.95	22.78	36.49
+WikiLM	45.07	16.53	33.50	50.94	26.21	42.85
+CCLM	50.39	16.67	35.88	59.98	28.17	48.93

TABLE 5.8: Effects of adding a large-scale language model to SMT system.

Finally, we compare the bilingual language models. On CoNLL-2013 test set the best results are achieved with OSM, whereas NNJM and the combination of the two gives slightly better results on CoNLL-2014 test set. Both OSM and NNJM have been shown to be effective for translation task with comparable results between these two joint models (Durrani et al., 2015). We actually confirm this for the GEC task.

The last row in Table 5.7 presents the scores achieved for the system that have been used in optimization experiments in the previous section. This system uses edit operation counts, WCLM and OSM.

5.4.3 Increasing the size of language model

We examine the impact of the size of a language model on the system performance. Until now we restricted monolingual data in our experiments to Wikipedia, which is a similar setting to the one used by Susanto et al. (2014). However, systems from the CoNLL-2014 shared task were free to use any publicly available data.

We use the English resources from CommonCrawl data (Buck et al., 2014) to build a web-scale language model. We trained a 5-gram language model using KenLM with heavy pruning, which resulted in a manageable 21G binary model[11].

Table 5.8 summarizes the results before and after adding the Wikipedia or Common Crawl language models to the baseline system with and without dense features. Recall that the *baseline* system uses NUCLE and Lang-8 NAIST as training data and a language model trained on the target-side of the data. *Dense* features include edit operation counts and OSM.

Adding a web-scale language model increases the precision of the system on both test sets, and thus allows to achieve the highest F-score. Recall, however, is slightly decreased with CCLM when no dense features are used.

The relative performance gain from the CCLM is higher for the system using dense features than for the bare system (+2.38% vs. +0.48% for the CoNLL-2013 test set). This suggests that a web-scale monolingual data allows the system to take the full advantage of the task-specific features.

[11]The language model is available for download from https://github.com/grammatical/baselines-emnlp2016

System	CoNLL-2013			CoNLL-2014		
	P	R	M^2	P	R	M^2
NUCLE+WikiLM	44.43	13.03	29.98	47.77	18.05	35.94
+Lang-8 NAIST	45.07	16.53	33.50	50.94	26.21	42.85
+Lang-8 WEB	43.58	16.09	32.49	53.56	29.59	46.09
NUCLE+CCLM	50.69	13.75	32.98	56.13	18.31	39.72
+Lang-8 NAIST	50.39	16.67	**35.88**	59.98	28.17	48.93
+Lang-8 WEB	48.73	17.13	35.60	61.74	30.51	**51.25**

TABLE 5.9: Effects of adding more parallel training data to SMT system.

5.4.4 Additional parallel data

The generation of translation options in a phrase-based SMT system is restricted to the phrases that appeared in training data, and thus are included in the phrase table. Besides the addition of a web-scale language model, we also examine the effects of large-scale training data for GEC translation models. The system is re-trained every time additional parallel data is added. We summarize the results in Table 5.9.

The additional large-scale in-domain data (Lang-8 WEB) originates from the social learner's platform Lang-8, same as the Lang-8 NAIST corpus, but consist of ca. 1.2M additional sentence pairs. In most experiments increasing the amount of parallel training data results in higher recall. While adding the larger Lang-8 corpus causes a respectable gain when used with the Wikipedia language model (+3.24%), the web-scale language model seems to cancel out part of the effect (+2.32%).

The system trained on Lang-8 WEB data with the Common Crawl language model gives the highest results on the CoNLL-2014 test set reported so far in this chapter. The lack of improvement for the CoNLL-2013 test set comes from the drop in recall. It can be caused by the fact that the test set is annotated only by a single annotator and thus may contain less diversified corrections than the other test set or even miss some corrections.

5.4.5 Incorporating additional out-of-domain data

It is a common scenario in SMT when less in-domain training data is available than out-of-domain data. The WikEd error corpus is out of domain for the ESL grammatical error correction, but it is ca. 20 times larger than the Lang-8 NAIST (24.4M vs 1.2M of sentence pairs). We examine the usefulness of this resource for GEC task.

We experiment with incorporating the out-of-domain data using the so-called phrase table fill-up method (Bisazza et al., 2011). It is a domain adaptation method that combines two translation models in the form of phrase tables to improve model coverage. Given the translation model learnt from in-domain data T_I which assigns a feature vector of scores $\phi_I(s,t)$ for each pair of source s and target t phrase, and the translation model estimated from out-domain data with feature vector $\phi_O(s,t)$, the fill-up model $T_{I \cup O}$ is defined as:

System	CoNLL-2013 P	R	M^2	CoNLL-2014 P	R	M^2
Lang-8 NAIST + WikiLM	45.07	16.53	33.50	50.94	26.21	42.85
+PT fill-up WikEd	44.00	17.48	**33.76**	50.46	29.02	**43.96**
Lang-8 NAIST + CCLM	50.39	16.67	**35.88**	59.98	28.17	48.93
+PT fill-up WikEd	44.52	18.43	34.70	57.93	31.73	**49.72**

TABLE 5.10: Results for the phrase table fill-up with out-of-domain data.

$$\forall (s,t) \in T_I \cup T_O :$$
$$\phi_{I \cup O}(s,t) = \begin{cases} (\phi_I(s,t), \mathrm{Ind}_I) & \text{if } (s,t) \in T_I \\ (\phi_O(s,t), \mathrm{Ind}_O) & \text{otherwise} \end{cases} .$$

The entries of $T_{I \cup O}$ correspond to the union of the two phrase tables. The scores are taken from the more reliable in-domain phrase table whenever possible. Ind is an additional feature which determines, which translation model the given phrase pair originates from, and has values of either $\exp 0$ or $\exp 1$. The weight assigned to Ind acts as a scaling factor for the out-of-domain phrases. It is optimized during the MERT tuning procedure together with the rest of the weights.

As presented in Table 5.10, filling-up the phrase table with WikEd data results in a performance gain on both test sets if a system is supplied with WikiLM. The method improves the recall at the little cost of precision. For the system using larger language model, a higher performance is only observed on the CoNLL-2014 test set.

5.5 Summary

This chapter described the application of statistical machine translation to automatic grammatical error correction. Despite the fact that this approach is among the most popular methods in GEC today, few papers that report results for the CoNLL-2014 test set seem to have exploited its full potential. An important aspect when training SMT systems that one needs to tune parameters towards the task evaluation metric seems to have been under-explored.

We have shown that a Moses SMT system trained on publicly available data achieves state-of-the-art results on the CoNLL-2014 test set. Proper M^2-based tuning contributed the most to these results. Better development sets and an optimization procedure using cross validation allowed to obtain more stable results.

With this tuning mechanism available, task-specific features have been explored that bring further improvements, such as edit operation counts of phrase pairs, and various language models. None of the explored features require complicated pipelines or re-ranking mechanisms. Instead they are a natural part of the log-linear model in phrase-based SMT. It is therefore quite easy to reproduce our results and the presented systems may serve as new baselines for automatic grammatical error correction.

Chapter 6

Discriminative Models for SMT-based Grammatical Error Correction

In the previous chapter, we have shown that an SMT system can achieve high performance in the GEC task and is able to correct a wide range of grammatical errors. This is possible due to the generative nature of the phrase-based statistical machine translation model. Historically, the most successful approaches in GEC were based on discriminative classification methods targeting specific error types. In this chapter, we study possibilities to improve the discriminative power of the SMT-based GEC system developed so far[1]. This is the first attempt to apply sparse features to automatic grammatical error correction. We show that a generative SMT system using task-specific discriminative features derived from correction patterns outperforms best reported results for any paradigm in GEC.

In Section 6.1, we introduce two discriminative models and show how they can be integrated with the log-linear model. Section 6.2 describes feature templates used to generate feature sets for each method. In Section 6.3, we present our experiments, which lead to new state-of-the-art results. A detailed evaluation and comparison of developed systems with results reported in the literature are presented in Section 6.4. We conclude in Section 6.5.

6.1 Discriminative models

The modern phrase-based statistical machine translation model draws on generative learning. A *generative method* generates data by estimating a distribution of the model parameters under specific assumptions and using this to predict unseen data. It assumes that the learned model represents the real model closely. A *discriminative method*, in contrast, models the boundary between possible outputs by depending on the observed data. It makes fewer assumptions on the distributions. For some prediction tasks, discriminative methods tend to outperform generative methods (Bouchard and Triggs, 2004, Ng and Jordan, 2002), especially when a sufficient quantity of training examples is available.

In phrase-based SMT, discriminative information is added in the form of discriminative lexicons to help address the challenges of sense disambiguation and morphological coherence

[1]The chapter based on findings described in Grundkiewicz and Junczys-Dowmunt (2015) and Grundkiewicz and Junczys-Dowmunt (2017), and presents extended work originally published in Junczys-Dowmunt and Grundkiewicz (2016).

(Tamchyna et al., 2016). We experiment with two different methods for including discriminative information into the phrase-based SMT system:

- A cost-sensitive multi-class classifier, which contributes to the log-linear model as a single feature function.
- Sparse features, which extend the log-linear model by a large number of binary feature functions.

Both methods are described in the following sections.

6.1.1 Discriminative classifier

Here, we use a discriminative classifier to score potential corrections of each source phrase during decoding. Classifier scores are used as a separate feature function in the log-linear model. It contributes to the overall score of each potential correction of the given source phrase similarly to all dense feature functions described earlier. The classifier is trained outside the training process of the SMT system on examples extracted from the phrase table using specifically-designed features.

In particular, we model the classifier output given the source sentence S, the target sentence T and their source-side and target-side phrasal segmentations[2], $(s_1, \ldots, s_n)$ and $(t_1, \ldots, t_m)$, respectively. The translation probability $p_{VW}(T|S)$ is the product of phrasal translation probabilities:

$$p_{VW}(T|S) \propto \prod_{(s_i,t_i)\in(T,S)} p_{VW}(t_i|s_i, S).$$

Target-side phrasal probabilities $p_{VW}(t_i|s_i, S)$ are conditioned on the source phrase (obtained from the word alignment between S and T) and the full source sentence. The latter provides an additional context, which is used to build a vector of classifier features.

The phrasal probabilities for each target phrase are defined using a weighted feature vector and normalized over possible corrections:

$$p_{VW}(t_i|s_i, S) = \frac{exp\big(w \cdot f_\phi(t_i, s_i, S)\big)}{\sum_{t'_i \in PT(s_i)} exp\big(w \cdot f_\phi(t'_i, s_i, S)\big)}.$$

The function $f_\phi(t_i, s_i, S)$ returns a vector of classifier features, and w is a vector of corresponding feature weights. Feature weights are estimated from the training data. The classifier prediction values are normalized over a set $PT(s_i)$ of possible phrasal translations for the source phrase s_i. The normalized exponential function (Bishop, 2006) is used, so that all values are in the range of $[0, 1]$ and sum up to 1.

The discriminative classification model contributes to the log-linear model as a single feature function $h_{VW} = p_{VW}$ with a single parameter weight:

$$\log p(T|S) = \sum_{i=1}^{N} \lambda_i \log h_i(T|S) + \lambda_{VW} \log h_{VW}(T|S).$$

[2] The word alignment is skipped for simplicity.

Hence, the SMT system has access only to the final score of the classifier, not to the classifier features.

The requirement for the classifier is to predict scores for all target phrases, which are too numerous to form a set of labels (or classes) of reasonable size[3]. We use the *cost-sensitive multi-class classifier* with Label Dependent Features (LDF) from Vowpal Wabbit[4] for that task.

In the cost-sensitive scenario (Sammut and Webb, 2011), examples are not assigned single labels. Instead, different costs for each label are used. In our case, while training the classifier, the positive examples representing the ground truth translation phrases get a cost of 0 and all negative examples (i.e. target phrases possible for the given source phrase, but not chosen) get a cost of 1. Due to label dependent features, no label is provided to the classifier[5]. Instead, the classifier features are designed to describe each target phrase alternative accurately, so that the classifier is able to distinguish between positive and negative examples. During prediction, in turn, m binary classifiers are used in one-against-all framework (Sammut and Webb, 2011) to produce the score.

6.1.2 Sparse features

We also tested a method for incorporating a discriminative component into our SMT-based GEC system, we use *sparse features*. Sparse features have been introduced to syntax-based and phrase-based statistical machine translation (Chiang et al., 2009, Hasler et al., 2012) leading to some improvements in translation quality. They are intended to make the model aware of specific phenomena through the use of discrete, and most commonly, binary values. In the case of grammatical error correction, for instance, we may want to design a binary feature, which is active if a particular determiner is replaced by another, e.g. *an* → *the*. Such a feature describes a "correction pattern", which may help discriminate between good and bad corrections.

Each sparse feature contributes to the log-linear model as a separate feature function $h_{CP,j}$ with its own weight λ_j:

$$\log p(T|S) = \sum_{i=1}^{N} \lambda_i \log h_i(T|S) + \sum_{j=1}^{M} \lambda_j \log h_{CP,j}(T|S).$$

Our sparse features are binary-valued, and indicate the existence of a correction pattern between the source and translation sentences, i.e.:

$$h_{CP,j}(T|S) = \mathbb{1}_{C(T,S)}(c_j) = \begin{cases} 1 & \text{if } c_j \in C(T,S) \\ 0 & \text{otherwise} \end{cases}.$$

C is the set of all observed correction patterns, and $C(T,S)$ is the set of patterns between the given sentences S and T. The total number of correction patterns is $|C| = M$.

The feature weights λ decide which features are useful for making corrections and which are not. The higher the absolute value of the weight λ is, the more significant correction pattern

[3] The number of unique target phrases in the phrase table trained on the NUCLE and Lang-8 NAIST corpora, filtered for the CoNLL test sets, and limited to the top 50 target phrases for each source phrase is ca. 913,000.

[4] https://github.com/JohnLangford/vowpal_wabbit

[5] Label dependent features and LDF format itself are described more thoroughly in Section 6.2.1.

becomes. Weights of sparse features are trained during tuning. The parameter tuning algorithm MERT (Och, 2003), which we used for tuning dense features, does not support a large number of sparse features, which can easily grow to hundred of thousands or millions. Instead, we use the MIRA (Cherry and Foster, 2012) algorithm and experiment with other algorithms, as described in Section 6.3.2

It should be noted, that no information from the translation model is used while generating sparse features. This differs from the discriminative classifier, which uses training examples obtained from the phrase table. Sparse features are extracted directly from a pair of the source and target sentences during decoding. The final set of sparse features is created during the tuning procedure and consists of correction patterns observed in the tuning set. This requires a larger development set than that needed for dense features, and fits to our NUCLE-based tuning procedure introduced in Section 5.4.1.3.

By adding a plethora of sparse feature functions, we improve the discriminative power of the translation model.

6.2 Feature templates

All features, which we use for discriminative training, are automatically constructed features based either on n-grams or on edit operations. Similar to features exploited for SMT, the discriminative features are not modeled with a specific language or error type in mind. They are language-independent in the sense that they are not build on top of linguistic phenomena and can be easily extracted for most languages.

This stands in contrast to the line of research using the classification approach, for which extensive feature engineering to GEC has been applied (De Felice and Pulman, 2008, Rozovskaya and Roth, 2014, Rozovskaya et al., 2014b). However, in Grundkiewicz and Junczys-Dowmunt (2015), we have shown that simple n-gram-based features can be as effective as linguistically-motivated heuristics in the case of article and preposition error correction. This has motivated us to use features in a similar fashion in SMT.

6.2.1 Label-dependent features

We make use of multi-label classification with *label-dependent* features provided by Vowpal Wabbit while creating features for discriminative classifiers. All classifier features are divided into two groups:

- Label-Independent (LI) features, which are generated for the source phrase and use additional context information from the source sentence.
- Label-Dependent (LD) features, which are generated for the particular target phrase using source phrases and the source sentence context, as well as target phrases.

The label-independent features are generated from the following templates:

- **Source phrase**: An indicator of the source phrase, i.e. the original phrase being corrected. It is created by concatenating either words or word classes within the source phrase.
- **Source words**: A set of words from the source phrase without order preservation.

- **Source window**: A set of words preceding or following the source phrase. The relative order in relation to the source phrase is preserved. Left and right context sizes do not exceed 3.

- **Word n-grams**: A set of word n-grams around the source phrase. All n-grams have lengths between two and four and include at least one word from the source phrase. We extract n-grams of words and automatic word classes.

For example, for the phrase «*new problem*» in the sentence «*Then a new problem comes out .*», we generate the following label-independent features on the word level:

```
sind=new_problem
sword=new
sword=problem
swin(-2)=then
swin=(-1)=a
swin(1)=comes
swin(2)=out
swin(3)=.
2gram(2)=a_new 2gram(3)=new_problem 2gram(4)=problem_comes
3gram(2)=then_a_new 3gram(3)=a_new_problem 3gram(4)=new_problem_comes
  3gram(5)=problem_comes_out
4gram(2)=<s>_then_a_new 4gram(3)=then_a_new_problem 4gram(4)=a_new_problem_comes
  4gram(5)=new_problem_comes_out 4gram(6)=problem_comes_out_.
```

The set of label-dependent features includes features obtained from the following templates:

- **Target phrase**: An indicator of the target phrase, i.e. the potential correction of the phrase source.

- **Edit operation counts**: A set of the counts of inserted, deleted and substituted words between the source and target phrase.

- **Edit operations**: A set of inserted, deleted or substituted tokens that took part in the edit of the source and target phrase. Edit operations are built from words or automatic word classes.

For example, for the edit pair («*new problem*», «*a new problem had*») for the sentence «*Then a new problem comes out .*», we generate the following label-dependent features on the word level:

```
tind=a_new_problem_had
ecount(del)=0
ecount(ins)=2
ecount(sub)=0
ecount(eql)=2
eops(ins)=a
eops(ins)=had
```

The process of extracting the final set of features for the classifier is as follows: firstly, for each source phrase s_i, we generate a set of label-independent (LI) features. Then, for each possible correction t_i of s_i provided by the phrase table PT, we generate a set of label-dependent (LD) features. Next, we create the cartesian product of the two feature sets LI $\times$ LD and add them as new features. All final features are binary.

Thanks to the cartesian product of the two namespaces, some desired features are created automatically. For example, **Source phrase** and **Target phrase** feature templates produce an indicator of the pair of the source and target phrase, or **Source window** and **Edit operations** templates create a set of edit operation features with the left/right context included.

6.2.2 Sparse edit operations

Introducing stateless dense features (Chapter 5, Section 5.2.1) based on finer-grained edit operations improved performance of the SMT system. We extend that idea by sparse features that describe specific correction patterns with and without context.

For each pair of the source and target sentences we extract all edits at token level based on the Levenshtein distance matrix. Each sparse feature contains the type of the basic edit operation (i.e. substitution, deletion or insertion) and orthographic forms of tokens that took part in the edit. A feature might additionally include the left and/or right context of a fixed size. Similarly as in the case of the features of the discriminative classifier, the context annotation comes from the erroneous source sentence, not from the corrected target sentence. We further investigate different source factors — elements taking part in the edit operation or appearing in the context can either be word forms (factor 0) or word classes (factor 1).

In particular, we use the following sparse feature types:

- **E0**: A set of basic edit operations on words, no context.
- **E1**: A set of basic edit operations on word classes, no context.
- **E0C10**: A set of edit operations on words with left/right context of maximum length 1 on words.
- **E1C11**: An edit operation on word classes with left/right context of maximum length 1 on word classes
- **E0C11**: An edit operation on words with left/right context of maximum length 1 on word classes.
- **E0C21**: An edit operation on words with left/right context of maximum length 2 on word classes.

Fo example, for the pair of the erroneous and corrected sentences («*Then a new problem comes out .*», «*Hence , a new problem surfaces .*»), we generate the following sparse features that model contextless edits (matches are omitted):

```
sub(Then,Hence)
ins(,)
sub(comes,surfaces)
del(out)
```

and sparse features with one-sided left or right or two-sided context:

```
<s>_sub(Then,Hence)
sub(Then,Hence)_a
Hence_ins(,)
ins(,)_a
problem_sub(comes,surfaces)
sub(comes,surfaces)_out
comes_del(out)
del(out)_.
<s>_sub(Then,Hence)_a
Hence_ins(,)_a
problem_sub(comes,surfaces)_out
comes_del(out)_.
```

When using sparse features with context, the contextless features are included.

6.3 Experiments

We compare the systems with discriminative components to our best dense-feature systems developed in Chapter 5. Both extensions are added on-top of best dense features, which we refer to as *best dense*. The *best dense* system features include edit operation counts, operation sequence model, and a language model built from the corrected sentences from training data. We report results for both additional language model sets: 5-gram LM on words and 9-gram LM on word classes, which are built respectively on Wikipedia texts (WikiLM) and Common Crawl data (CCLM). We also compare to the *baseline* system, which uses no dense features.

As training data, we use Lang-8 NAIST and NUCLE corpora. We follow the 4-fold cross validation described in Section 5.4.1.3, which uses NUCLE as a development set. Tuning is performed in 20 separate steps in total, and the final results are obtained with averaged parameter vector weights. The larger development set facilitates tuning for large number of discriminative features, especially in the case of sparse features.

6.3.1 Multi-class discriminative classifier

To train the classifier, we firstly generate label-independent and label-dependent features for training data on the basis of the phrase table from the SMT system trained on Lang-8 NAIST and NUCLE corpora. The maximum number of target phrases for each source phrase is limited to 50. While generating features for the classifier, we use leave-one-out technique to avoid overfitting, i.e. for each source phrase, we skip a random target phrase and no label-dependent features are generated for that.

Next, the classifier is trained using cost-sensitive multi-class classification with label-dependent features provided by the Vowpal Wabbit toolkit. We use the logistic cost function and enable

	CoNLL-2013			
System	Acc	P	R	$F_{0.5}$
VW_{tok}	94.63	58.77	7.07	23.88
VW_{wc}	94.70	63.43	8.42	27.49
$VW_{edits,wc}$	94.77	71.97	7.81	27.23

TABLE 6.1: The classifier performance on CoNLL-2013 with different feature sets.

quadratic features, which perform the cartesian product between label-independent and label-dependent features described in Section 6.2.1[6]. We train the model for 2 epochs, and then the trained model is added to the Moses configuration as an additional feature function.

A single classifier model trained on the Lang-8 NAIST corpus is used for each of the 4 folds in the cross validation scheme. For the final model, we additionally resume training on NUCLE for one epoch. Vowpal Wabbit supports continued training by saving extra learning parameters. We use the MERT tuning algorithm to tune weights for all feature functions.

6.3.1.1 External evaluation

We can evaluate the classifier performance independently of the SMT system by measuring how effectively it predicts correct target phrases on unseen data. For that purpose, we firstly generate all features from the CoNLL-2013 test set as during the training, not using the leave-one-out method. Then, the trained classifier is run on the generated features to produce predictions. We use the WAS evaluation scheme (introduced in Chapter 4, Section 4.2.1) for evaluation: the positive class consists of that phrases which require to be corrected according to the gold standard annotations, other phrases (which form the majority) belongs to the negative class. Results for three different feature sets are presented in Table 6.1.

The first feature set VW_{tok} uses all features described earlier except edit operations, and only lexical word forms are used for generation. VW_{wc} additionally uses features on word classes. The last feature set also includes edit operations. We refer to the set as $VW_{edits,wc}$.

All three feature sets improved accuracy over the baseline of 94.47%, which is calculated for unchanged input. Improvements of accuracy are minor (0.16–0.30%). In general, precision of the classifiers is relatively high, but recall is low. Each of the consecutive feature sets significantly improves precision. However, adding edit operation features slightly decreases recall, and the best $F_{0.5}$ score is achieved for VW_{wc}.

6.3.1.2 Results for VW feature function

In final experiments, we use two feature sets trained on word forms and word-class features without (VW_{wc}) or with ($VW_{edits,wc}$) edit operation features. Results are presented in Table 6.2.

When the Wikipedia language model is used, the discriminative feature function gives small improvements in $F_{0.5}$ score for both test sets. The exception is VW_{wc} on the CoNLL 2013 test set. Surprisingly, the system with $VW_{edits,wc}$ feature set shows better results than the system that uses the feature set without edit operations on the CoNLL 2013 test set, even though its

[6] The complete command used to train VW is: `vw -hash all -loss_function logistic -noconstant -b 26 -q st -csoaa_ldf mc -f model.vw`. Option `-b` sets the bit precision, `-q st` performs the carthesian product between two namespaces, `-noconstant` eliminates the constant feature that exists by default in VW, and `-hash all` hashes all feature identifiers increasing the performance time.

System	CoNLL-2013 P	R	M^2	CoNLL-2014 P	R	M^2
Best dense+WikiLM	45.07	16.53	33.50	50.94	26.21	42.85
+VW_{wc}	45.15	16.15	33.22	51.50	26.02	**43.07**
+$VW_{edits,wc}$	45.06	16.73	**33.66**	51.02	26.34	**42.97**
Best dense+CCLM	50.39	16.67	**35.88**	59.98	28.17	**48.93**
+VW_{wc}	49.53	16.87	35.71	59.73	28.23	48.83
+$VW_{edits,wc}$	51.18	16.30	35.84	59.83	27.50	48.44

TABLE 6.2: The SMT system performance with best VW feature function.

Optimizer	2013	2014
MERT	33.50	42.85
PRO	33.68	40.34
Mira	29.19	34.13
`-model-bg`	31.06	43.88
`-D` 0.001	33.86	42.91

TABLE 6.3: Tuning with different optimizers with dense features.

ability to predict the correct target phrases in the external evaluation were worse on that test set. Improvements on the test set from 2014 are +0.23% and +0.13%.

Improvements cannot be confirmed for the system using a web-scale language model. Adding the discriminative classifier decreases the F-score on both test sets (from -0.04% to -0.49%). Further investigation showed that the largest weight has been assigned to the language model during tuning, whereas the VW feature function recieved one of the lowest scores.

We see two issues with the presented method of classifier integration into the SMT system in the context of grammatical error correction. First, the classifier is trained on the same training data as the SMT system. This limits the discriminative power of the classifier as it acquires the same context information to discriminate between possible corrections as SMT already has. Rozovskaya and Roth (2016) showed that a simple pipeline of classifiers and an SMT system can significantly improve correction if the two frameworks are trained on different data[7].

The second issue is that for all classifier predictions we use a single weight in the log-linear model. If the classifier is poor only in the correction of, for example, a specific group/type of errors, the tuning algorithm will likely assign a lower weight for the feature function. This will decrease the overall contribution of the classifier, including the correction of errors for which the classifier possibly achieves a good accuracy. The latter issue is avoided when sparse features are used.

6.3.2 Tuning sparse features

Tuning sparse features according to the M^2 metric poses some problems. The MERT algorithm (Och, 2003) included in Moses cannot handle effectively parameter tuning with sparse feature

[7] Although, the SMT system used in that experiment was not tuned according to the evaluation metric and it is not clear if the same improvement would be achievable for the system tuned towards M^2.

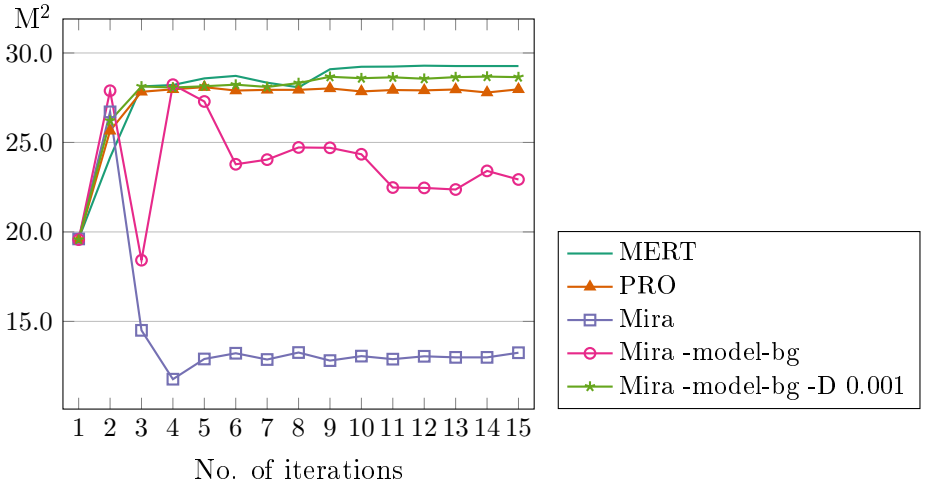


FIGURE 6.1: Results per iteration on development set (4-th NUCLE fold) for the system with dense features.

weights and one of the other optimizers has to be used. MERT is able to handle only the order of tens of features and is prone to overfitting.

We first experimented with both, PRO (Hopkins and May, 2011) and Batch Mira (Cherry and Foster, 2012), for the dense features only. We found PRO and Batch Mira with standard settings to either severely underperform in comparison to MERT or to suffer from instability with regard to different test sets (Table 6.3).

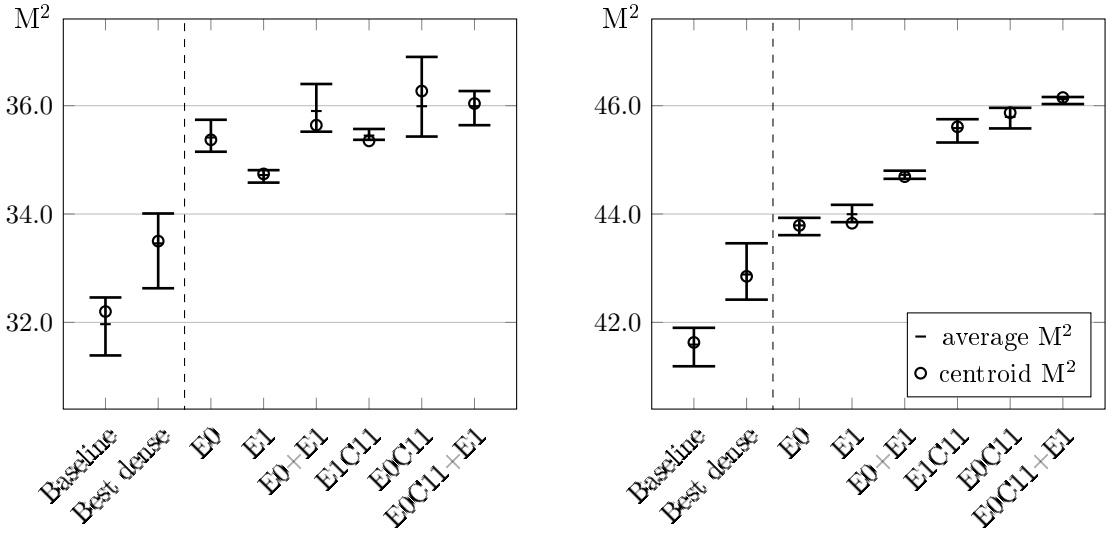
Experiments with Mira hyper-parameters allow to counter these effects. We first change the background BLEU approximation method in Batch Mira to use model-best hypotheses (`-model-bg`) which seems to produce more satisfactory results. Inspecting the tuning process, however, reveals problems with this setting. Figure 6.1 documents how instable the tuning process with Mira is across iterations. The best result is reached after only three iterations. In a setting with sparse features this would result in only a small set of weighted sparse features.

Mira's behavior seems to stabilize across iterations, when we set the background corpus decay rate to 0.001 (`-D 0.001`), resulting in a sentence-level approximation of M^2. At this point it is not quite clear why this is required. While PRO's behavior is more stable during tuning, results on the test sets are subpar. It seems that no comparable hyper-parameter settings exist for PRO.

6.3.3 Sparse feature sets

Figure 6.2 and Table 6.4 summarize the results for experiments with sparse features. All sparse feature types are added on-top of our best dense-features systems. As before, we average sparse feature weights across folds and multiple tuning runs.

All systems with sparse features outperform the system with dense features. On both test sets we can see significant improvements when including edit-based sparse features, and the performance increases even more when source context is added. The CoNLL-2013 test set contains annotations from only one annotator and is strongly biased towards high precision which might explain the greater instability. It appears that sparse features with context, where surface forms and word-classes are mixed, allow for the best fine-tuning. The highest improvement for



(A) CoNLL-2013 test set (B) CoNLL-2014 test set

FIGURE 6.2: Results for different sparse features sets.

	CoNLL-2013			CoNLL-2014		
System	P	R	M^2	P	R	M^2
Best dense	45.07	16.53	33.50	50.94	26.21	42.85
E0	47.92	17.28	35.37	52.95	25.88	43.79
E1	46.08	17.51	34.74	51.76	27.17	43.83
E0+E1	48.43	17.34	35.64	54.35	26.12	44.69
E1C11	50.73	15.98	35.35	57.27	25.14	45.61
E0C11	51.10	16.79	**36.27**	57.28	25.53	45.87
E0C11+E1	51.55	16.35	36.04	58.61	24.89	**46.12**

TABLE 6.4: The SMT system performance with various sparse edit operations.

the CoNLL-2014 test set is reported for E0C11+E1, which increases precision by as much as +7.67%.

Appendix B lists top-weighted sparse features for E0C11+E1.

6.3.4 Additional data

Table 6.5 summarizes the best results reported for dense and sparse features before and after adding more training and/or language model data. While our sparse features cause a respectable gain when used with the smaller language model (+2.54% and +3.10% on the CoNLL-2013 and CoNLL-2014 test set respectively), the web-scale language model seems to cancel out part of the effect (+1.04% and +0.56%). A similar situation occurs for the test set from 2014 when more parallel training data is added instead. However, using a larger training data set on top of the CCLM seems to allow for better use of correction patterns, and a substantial gain is visible for the system using sparse features.

System	CoNLL-2013 P	R	M^2	CoNLL-2014 P	R	M^2
Best dense	45.07	16.53	33.50	50.94	26.21	42.85
+Lang-8 WEB	43.58	16.09	32.49	53.56	29.59	46.09
+CCLM	50.39	16.67	35.88	59.98	28.17	48.93
+Lang-8 WEB	48.73	17.13	35.60	61.74	30.51	51.25
Best sparse	51.55	16.35	36.04	57.99	25.11	45.95
+Lang-8 WEB	49.38	16.09	34.93	58.57	27.11	47.54
+CCLM	51.50	17.31	36.92	61.27	27.98	49.49
+Lang-8 WEB	52.47	17.83	**37.78**	63.52	30.49	**52.21**

TABLE 6.5: Effects of adding a large-scale language model to SMT system.

6.4 Evaluation

In this section, we compare our best dense and sparse systems with other results published on the CoNLL-2014 test set, and confront them with the upper-bound estimated by Bryant and Ng (2015).

6.4.1 Comparison with other systems

In Table 6.6 we collect the published results for the CoNLL-2014 test set in comparison to our results. Among the top-three positioned systems during the CoNLL-2014 shared task, two submissions — CAMB Felice et al. (2014) and AMU Junczys-Dowmunt and Grundkiewicz (2014) — were partially or fully based on statistical machine translation. The second system, CUUI Rozovskaya et al. (2014a), was a classifier-based approach. The AMU system was our own contribution that introduced some of the concepts discussed in this thesis. Later analysis revealed that our submission had an incorrectly filtered language model that was missing a significant number of possible entries. Nonetheless, our SMT system with M^2 tuning trained on Lang-8 NAIST data and using Wikipedia language model already beats the top CUUI system (41.63% vs. 37.33%).

Our SMT system with additional task-specific features outperforms most of the more recent results (the middle section of Table 6.6), including the GEC system combinations published after the shared task by Susanto et al. (2014). Many of recent results also rely on MT systems with new features (Chollampatt et al., 2016b) or n-best list re-ranking methods (Hoang et al., 2016, Mizumoto and Matsumoto, 2016, Yuan et al., 2016). However, most of the improvement over the CoNLL-2014 shared task of these works originates from using the parameter tuning tools we introduced in Junczys-Dowmunt and Grundkiewicz (2014). Improvements are also reported by systems relying on neural sequence-to-sequence models (Xie et al., 2016, Yuan and Briscoe, 2016).

Participants of the CoNLL-2014 shared task were allowed to use additional publicly available data. Xie et al. (2016), who use a neural network-based approach to GEC, use data from the Lang-8 corpus and combine their model with an n-gram language model trained on web-scale Common Crawl data. The authors report a score of 40.56% M^2. Adding a similar-size language model (CCLM) to our system results in 48.93% M^2.

System	CoNLL-2014 P	R	M^2
The CoNLL-2014 shared task			
AMU* (Junczys-Dowmunt and Grundkiewicz, 2014)	41.62	21.40	35.01
CUUI (Rozovskaya et al., 2014a)	41.78	24.88	36.79
CAMB (Felice et al., 2014)	39.71	30.10	37.33
Recently published results			
Yuan et al. (2016)	–	–	38.08
Susanto et al. (2014)	53.55	19.14	39.39
Yuan and Briscoe (2016)	–	–	39.90
Mizumoto and Matsumoto (2016)	45.80	26.60	40.00
Xie et al. (2016)	49.24	23.77	40.56
Hoang et al. (2016)	50.56	22.68	40.58
Chollampatt et al. (2016b)	52.34	23.07	41.75
Rozovskaya and Roth (2016)*	60.17	25.64	47.40
This work			
Baseline	48.97	26.03	41.63
+CCLM	58.91	25.05	46.37
Best dense	50.94	26.21	42.85
+CCLM	59.98	28.17	48.93
+Lang-8 WEB*	61.74	30.51	**51.25**
Best sparse	57.99	25.11	45.95
+CCLM	61.27	27.98	49.49
+Lang-8 WEB*	63.52	30.49	**52.21**

TABLE 6.6: Comparison of our SMT-based GEC systems with the published results.

As mentioned before, Rozovskaya and Roth (2016) trained their systems on Lang-8 WEB data that has been collected by us. Since this data has not been made officially available, we treat it as non-public and mark all systems in Table 6.6 that use it with a star. This makes it difficult to put their results in relation with previously published work, but we can at least provide a comparison for our systems. As our strongest MT-only systems trained on public data already outperform the pipelined approaches from Rozovskaya and Roth (2016) (49.49% vs. 47.40%), it is unsurprising that adding more error-corrected parallel data results in an even wider gap (52.21% vs 47.40%). We can assume that this gap would persist also in the case when only public data were used.

It should be emphasized that bare-bones Moses trained on publicly available data rivals the best reported systems on the CoNLL-2014 test set. The *baseline* system using the CoNLL-2013 test set as a development set (Figure 5.1b) already achieves an F-score of 40.66%, so the largest improvement is achieved alone due to M^2-based tuning. Better tuning sets, task-specific dense and sparse features, and more data increase that performance advantage.

6.4.2 Upper-bound for the task

Bryant and Ng (2015) extended the CoNLL-2014 test set with additional annotations from originally two to new ten annotators. We report results obtained for this resource in Table 6.7.

System	GEC-10 P	GEC-10 R	GEC-10 M^2
AMU (Junczys-Dowmunt and Grundkiewicz, 2014)	58.53	31.68	50.05
CUUI (Rozovskaya et al., 2014a)	59.11	34.43	51.70
CAMB (Felice et al., 2014)	59.07	40.64	54.16
Baseline	69.22	37.00	58.95
+CCLM	76.66	36.39	62.77
Best dense	71.11	37.44	60.27
+CCLM	79.76	39.52	66.27
+Lang-8 WEB	80.50	42.29	68.18
Best sparse	76.48	35.99	62.43
+CCLM	80.57	39.74	66.83
+Lang-8 WEB	81.43	41.92	68.52
Avg. human annotator (Bryant and Ng, 2015)	–	–	≥72.58

TABLE 6.7: Evaluation of the SMT-based systems on the GEC-10 test set.

The higher M^2 on the GEC-10 test set is due to the fact that systems generally achieve higher recall (and precision) as it has been annotated by ten annotators. When several annotators are used as the gold standard, the M^2 scorer chooses whichever annotator for the given sentence produces the highest F-score.

According to Bryant and Ng (2015), human annotators seem to reach on average 72.58% M^2, which can be seen as an upper-bound for the task. The authors re-assessed the CoNLL-2014 systems with this extended test set and estimated that the top 3 teams perform, on average, between 67–73% as reliably as this human upper bound. The value has been calculated by excluding one annotator from the set of 10 annotators and using annotations from remaining 9 annotators as references. Our best SMT system with sparse features is the closest one to approach this upper-bound, reaching ca. 90.8% of it[8].

6.5 Summary

This chapter examined two methods for incorporating discriminative components into the log-linear model of the phrase-based statistical machine translation in order to improve the quality of grammatical error correction. A number of binary-valued feature functions in the form of sparse features based on correction patterns has shown to be more effective than a single feature function based on a cost-sensitive multi-class discriminative classifier. Our system with task-specific sparse features achieved the new state-of-the-art for ESL grammatical error correction at the CoNLL 2014 test set.

Scripts, system outputs, data sets, and trained models have been made publicly available for better reproducibility[9].

[8] The value is calculated as the division of the average system-vs-human and the average human-vs-human performance (i.e. 72.58%).

[9] https://github.com/grammatical/baselines-emnlp2016

Chapter 7

Summary

In this dissertation we have presented research investigating the application of statistical machine translation to the task of automated grammatical error correction. For this purpose, a number of methods and algorithms have been described.

In Chapter 2 we defined the GEC task and presented historical and recent approaches to the task. In Chapter 3 we described error corpora and monolingual data, and introduced a new resource named the WikEd Error Corpus. In order to find the optimal automatic metric for the task, we evaluated relevant metrics in terms of correlation with human judgment by conducting the first published large-scale human evaluation of GEC systems in Chapter 4. Next, Chapter 5 presented our contribution to GEC using machine translation, introducing parameter tuning towards a task-specific evaluation metric and task-specific dense features. In Chapter 6 we enhanced the discriminative power of the SMT-based GEC systems with sparse features based on correction patterns.

7.1 Contributions

Within the reported research we created the WikEd Error Corpus[1] — a publicly available large corpus of corrective Wikipedia edits of various types. Advantages of WikEd are its size (ca. 50 million of sentences) and permissive license.

We have successfully adapted methods from human evaluation campaigns from the Workshop on Machine Translation to automatic grammatical error correction. We provided analysis of correlation between the standard metrics in GEC and human judgment and showed that the commonly used parameters for standard metrics in the shared task correlate strongly but still may not be optimal. The collected and produced data has been made available[2] and already proved to be useful for other researchers, e.g. Napoles et al. (2016), Sakaguchi et al. (2016), Yuan et al. (2016).

Despite the fact that statistical machine translation approaches are among the most popular methods in automatic grammatical error correction, few papers that report results for the CoNLL-2014 test set seem to have exploited its full potential. In particular, the process of tuning parameters towards the task evaluation metric has been under-explored so far.

We have shown that a pure SMT-based system actually outperforms the best reported results for any paradigm in GEC provided that correct parameter tuning is performed. With this tuning

[1] https://github.com/snukky/wikiedits
[2] https://github.com/grammatical/evaluation

mechanism available, task-specific features have been explored that bring further significant improvements, putting phrase-based SMT ahead of other approaches by a large margin. None of the explored features require complicated pipelines or re-ranking mechanisms. Instead, they form a natural part of the log-linear model in phrase-based SMT. It is therefore quite easy to reproduce our results and the presented systems may serve as new baselines for automatic grammatical error correction. Our systems and scripts have been made available for better reproducibility[3].

7.2 Future research

Research presented in this dissertation provides suggestions for future work on automated grammatical error correction. A deeper study on the following topics might be valuable for the field:

- Exploration of different methods for adaptation of noisy edit data (e.g. the WikEd corpus, the Lang-8 corpus) to grammatical error correction. A comparison of domain-adapted data with artificially generated data would be valuable.
- Development of a better system-level evaluation metric and a meaningful sentence-level metric. Recent studies shift from edit-based metrics towards fluency-based metrics (Napoles et al., 2016, 2017).
- Quantitative analysis of corrections produced by the SMT-based system with regard to the specific error types. The weakness of MT systems are spelling errors, which may be addressed by encoding spell checker suggestions into a word lattice (Dyer et al., 2008), by transliteration techniques (Durrani et al., 2014) or subword units (Sennrich et al., 2016).
- Application of neural machine translation, which recently outperforms many phrase-based SMT models for traditional machine translation. Recently reported results using the neural approach are still below the baselines introduced in this work.
- Examination on the effectiveness of the developed techniques apply to texts written by native-speakers and to other languages, especially highly inflected languages with free word order.

[3] https://github.com/grammatical/baselines-emnlp2016

Appendix A

Error types in the NUCLE corpus

Type	Description	Example
Vt	Verb tense	Medical technology during that time [is → was] not advanced enough to cure him.
Vm	Verb modal	Although the problem [would → may] not be serious, people [would → might] still be afraid.
V0	Missing verb	However, there are also a great number of people [who → who are] against this technology.
Vform	Verb form	A study in 2010 [shown → showed] that patients recover faster when surrounded by family members.
SVA	Subject-verb agreement	The benefits of disclosing genetic risk information [outweighs → outweigh] the costs.
ArtOrDet	Article or determiner	It is obvious to see that [internet → the internet] saves people time and also connects people globally.
Nn	Noun number	A carrier may consider not having any [child → children] after getting married.
Npos	Noun possessive	Someone should tell the [carriers → carrier's] relatives about the genetic problem.
Pform	Pronoun form	A couple should run a few tests to see if [their → they] have any genetic diseases beforehand.
Pref	Pronoun reference	It is everyone's duty to ensure that [he or she → they] undergo regular health checks.
Prep	Preposition	This essay will [discuss about → discuss] whether a carrier should tell his relatives or not.
Wci	Wrong collocation/idiom	Early examination is [healthy → advisable] and will cast away unwanted doubts.
Wa	Acronyms	After [WOWII → World War II], the population of China decreased rapidly.
Wform	Word form	The sense of [guilty → guilt] can be more than expected.

Type	**Description**	**Example**
Wtone	Tone (formal/informal)	[It's → It is] our family and relatives that bring us up.
Srun	Run-on sentences, comma splices	The issue is highly [debatable, a → debatable. A] genetic risk could come from either side of the family.
Smod	Dangling modifiers	[Undeniable, → It is undeniable that] it becomes addictive when we spend more time socializing virtually.
Spar	Parallelism	We must pay attention to this information and [assisting → assist] those who are at risk.
Sfrag	Sentence fragment	However, from the ethical point of view.
Ssub	Subordinate clause	This is an issue [needs → that needs] to be addressed.
WOinc	Incorrect word order	[Someone having what kind of disease → What kind of disease someone has] is a matter of their own privacy.
WOadv	Incorrect adjective/adverb order	In conclusion, [personally I → I personally] feel that it is important to tell one's family members.
Trans	Linking words/phrases	It is sometimes hard to find [out → out if] one has this disease.
Mec	Spelling, punctuation, capitalization, etc.	This knowledge [maybe relavant → may be relevant] to them.
Rloc-	Redundancy	It is up to the [patient's own choice → patient] to disclose information.
Cit	Citation	Poor citation practice.
Others	Other errors	An error that does not fit into any other category but can still be corrected.
Um	Unclear meaning	Genetic disease has a close relationship with the born gene. (i.e., no correction possible without further clarification.)

TABLE A.1: Error types in the NUCLE corpus. Table adapted from Dahlmeier et al. 2013.

Appendix B

Sparse edit operation weights

Num.	Correction pattern	Weight
1	left(«<s>»)_ins(«the»)	0.154720
2	sub(«168»,«12»)	0.152366
3	ins(«the»)_right(«37»)	0.145961
4	sub(«124»,«12»)	0.144634
5	ins(«the»)_right(«160»)	0.135088
6	sub(«168»,«180»)	0.133227
7	left(«178»)_ins(«,»)	0.128886
8	sub(«had»,«have»)	0.104859
9	left(«<s>»)_sub(«Government»,«the government»)	0.103469
10	sub(«Government»,«the government»)	0.103273
11	sub(«G»,«161»)	0.098490
12	sub(«VHTR»,«vhtr»)	0.098490
13	ins(«97»)	0.097074
14	del(«the»)	0.096451
15	del(«the»)_right(«94»)	0.096307
16	del(«97»)	0.095509
17	del(«of»)	0.094942
18	left(«125»)_sub(«reactor»,«reactors»)	0.093740
19	left(«133»)_ins(«,»)_right(«12»)	0.090024
20	sub(«old-aged»,«aged»)_right(«2»)	0.090007
21	del(«on»)	0.087010
22	sub(«153»,«69»)	0.086614
23	sub(«has»,«have»)	0.083968
24	sub(«had»,«has»)	0.083948
25	ins(«168»)	0.083487
26	ins(«an»)	0.082840
27	left(«63»)_ins(«the»)_right(«160»)	0.081695
28	ins(«83»)	0.081025
29	sub(«is»,«was»)	0.077694
30	sub(«124»,«168»)	0.076637

TABLE B.1: Sparse edit operations with highest weights assigned by the MIRA algorithm. Correction patterns generated with E0C11+E1 template.

Abbreviations

Acc	Accuracy
BLEU	BiLingual Evaluation Understudy
CSS	Context-Sensitive Spelling
CLC	Cambridge Learner Corpus
CoNLL	Conference on Natural Language Learning
ESL	English as a Second Language
ER	Error Rate
F_β	F_β-score
FCE	First Certificate in English
FN	False Negative
FP	False Positive
GEC	Grammatical Error Correction
HOO	Helping Our Own
L1	the first language of the writer
L2	the second language of the writer
LM	Language Model
M^2	MaxMatch metric
METEOR	Metric for Evaluation of Translation with Explicit ORdering
MERT	Minimum Error Rate Training
ML	Machine Learning
MT	Machine Translation
NNJM	Neural Network Joint Model
NMT	Neural Machine Translation
NLP	Natural Language Processing
NUCLE	National University of Singapore Corpus of Learner English
SMT	Statistical Machine Translation
OSM	Operation Sequence Model
P	Precision
POS	Part Of Speech
PBSMT	Phrase-Based Statistical Machine Translation

R	Recall
SER	Sentence Error Rate
TN	True Negative
TP	True Positive
WC	automatic Word Class
WAcc	Weighted Accuracy
WAS	Writer-Annotator-System
WER	Word Error Rate
WMT	Workshop on Machine Translation
VW	Vowpal Wabbit

Symbols

General

p	probability
p'	pseudo-probability
d	distance function
$\mathbb{1}_A$	indicator function of a set A
$\bar{v}$	mean of a vector v

Evaluation

I	number of sentences in a test set
N	number of systems to be evaluated
M	maximum number of sentences presented to the evaluator
O_i	system ouput
E	set of system outputs
U_i	multiset of multiplicities
κ	Cohen's κ coefficient
r_i	system rank
ρ	Spearman's rank correlation
r	Pearson's correlation co-efficient

Machine translation

S, T	sequence of tokens
s_i, t_i	i-th token
$\hat{T}$	suggested correction sequence
h_i	feature function
λ_i	feature function weight
ϕ	feature vector

List of Tables

List of Figures

Bibliography

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*, pages 32–41, Atlanta, Georgia. Association for Computational Linguistics.

Atwell, E. S. (1987). How to detect grammatical errors in a text without parsing it. In *Proceedings of the Third Conference on European Chapter of the Association for Computational Linguistics*, EACL '87, pages 38–45, Stroudsburg, PA, USA. Association for Computational Linguistics.

Atwell, E. S. and Elliot, S. (1987). Dealing with ill-formed english text. *The Computational Analysis of English: A Corpus-Based Approach*, pages 120–138.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bassil, Y. and Alwani, M. (2012). Post-editing error correction algorithm for speech recognition using bing spelling suggestion. *arXiv preprint arXiv:1203.5255*.

Bender, E. M., Flickinger, D., Oepen, S., Walsh, A., and Baldwin, T. (2004). Arboretum: Using a precision grammar for grammar checking in call. In *In Proceedings of the InSTIL/ICALL Symposium: NLP and Speech Technologies in Advanced Language Learning Systems*, pages 83–86.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Bisazza, A., Ruiz, N., and Federico, M. (2011). Fill-up versus interpolation methods for phrase-based SMT adaptation. In *International Workshop on Spoken Language Translation (IWSLT) 2011*.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

Bojar, O., Buck, C., Callison-Burch, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., Soricut, R., and Specia, L. (2013). Findings of the 2013 Workshop on Statistical Machine Translation. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, pages 1–44. ACL.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H., et al. (2014). Findings of the 2014 Workshop on Statistical

Machine Translation. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 12–58. ACL.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M., Verspoor, K., and Zampieri, M. (2016). Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 131–198, Berlin, Germany. Association for Computational Linguistics.

Bojar, O., Ercegovčević, M., Popel, M., and Zaidan, O. (2011). A grain of salt for the WMT manual evaluation. In *Proc. of the Sixth Workshop on Statistical Machine Translation*, pages 1–11, Edinburgh, Scotland. ACL.

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*, pages 43–48, Baltimore, Maryland. Association for Computational Linguistics.

Bouchard, G. and Triggs, B. (2004). The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT'04)*, pages 721–728.

Brants, T. and Franz, A. (2006). Web 1t 5-gram version 1.

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 the 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Stroudsburg, USA. Association for Computational Linguistics.

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479.

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)*, pages 697–707, Beijing, China. Association for Computational Linguistics.

Buck, C., Heafield, K., and van Ooyen, B. (2014). N-gram counts and language models from the Common Crawl. In *Proceedings of the Language Resources and Evaluation Conference*, pages 3579–3584, Reykjavík, Iceland.

Buys, J. and van der Merwe, B. (2013). A tree transducer model for grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–51, Sofia, Bulgaria. Association for Computational Linguistics.

Cahill, A., Madnani, N., Tetreault, J., and Napolitano, D. (2013). Robust systems for preposition error correction using wikipedia revisions. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 507–517, Atlanta, Georgia. Association for Computational Linguistics.

Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C., and Schroeder, J. (2008). Further meta-evaluation of machine translation. In *Proc. of the Third Workshop on Statistical Machine Translation*, pages 70–106. ACL.

Cettolo, M., Bertoldi, N., and Federico, M. (2011). Methods for smoothing the optimizer instability in SMT. In *MT Summit XIII: the Thirteenth Machine Translation Summit*, pages 32–39.

Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cherry, C. and Foster, G. (2012). Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Stroudsburg, USA. Association for Computational Linguistics.

Chiang, D., Knight, K., and Wang, W. (2009). 11,001 new features for statistical machine translation. In *Proceedings of human language technologies: The 2009 annual conference of the north american chapter of the association for computational linguistics*, pages 218–226. Association for Computational Linguistics.

Chodorow, M., Dickinson, M., Israel, R., and Tetreault, J. R. (2012). Problems in evaluating grammatical error detection systems. In *Proceedings of COLING 2012*, pages 611–628.

Chodorow, M. and Leacock, C. (2000). An unsupervised method for detecting grammatical errors. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 140–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chodorow, M., Tetreault, J. R., and Han, N.-R. (2007). Detection of grammatical errors involving prepositions. In *Proceedings of the fourth ACL-SIGSEM workshop on prepositions*, pages 25–30. Association for Computational Linguistics.

Chollampatt, S., Hoang, D. T., and Ng, H. T. (2016a). Adapting grammatical error correction based on the native language of writers with neural network joint models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1901–1911, Austin, Texas. Association for Computational Linguistics.

Chollampatt, S., Taghipour, K., and Ng, H. T. (2016b). Neural network translation models for grammatical error correction. *arXiv preprint arXiv:1606.00189*.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. (2011). Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT '11, pages 176–181, Stroudsburg, USA. Association for Computational Linguistics.

Cohen, J. (1960a). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*.

Cohen, J. (1960b). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37.

Crysmann, B., Bertomeu, N., Adolphs, P., Flickinger, D., and Klüwer, T. (2008). Hybrid processing for grammar and style checking. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 153–160. Association for Computational Linguistics.

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-Volume 1*, pages 915–923. Association for Computational Linguistics.

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*, EMNLP-CoNLL '12, pages 568–578. Association for Computational Linguistics.

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*, pages 568–572. Association for Computational Linguistics.

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*, pages 216–224. Association for Computational Linguistics.

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*, pages 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*, pages 54–62. Association for Computational Linguistics.

Dale, R. and Kilgarriff, A. (2010). Helping our own: Text massaging for computational linguistics as a new shared task. In *Proceedings of the 6th International Natural Language Generation Conference*, INLG '10, pages 263–267, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dale, R. and Kilgarriff, A. (2011). Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242–249. Association for Computational Linguistics.

Daudaravicius, V. (2015). Automated evaluation of scientific writing: Aesw shared task proposal. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 56–63.

De Felice, R. and Pulman, S. (2007). Automatically acquiring models of preposition use. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, pages 45–50, Prague, Czech Republic. Association for Computational Linguistics.

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 - Volume 1*, COLING '08, pages 169–176, Stroudsburg, PA, USA. Association for Computational Linguistics.

Denkowski, M. and Lavie, A. (2011). Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proc. of the EMNLP 2011 Workshop on Statistical Machine Translation*.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Dini, L. and Malnati, G. (1993). Weak constraints and preference rules. *Studies in Machine Translation and Natural Language Processing*, pages 75–90.

Douglas, S. and Dale, R. (1992). Towards robust PATR. In *Proceedings of the 14th conference on Computational linguistics — Volume 2*, pages 468–474.

Durrani, N., Fraser, A., Schmid, H., Hoang, H., and Koehn, P. (2013). Can Markov Models Over Minimal Translation Units Help Phrase-Based SMT? In *ACL (2)*, pages 399–405. The Association for Computer Linguistics.

Durrani, N., Sajjad, H., Hoang, H., and Koehn, P. (2014). Integrating an unsupervised transliteration model into statistical machine translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 148–153, Gothenburg, Sweden. Association for Computational Linguistics.

Durrani, N., Sajjad, H., Shafiq Joty, A. A., and Vogel, S. (2015). Using joint models for domain adaptation in statistical machine translation. *Proceedings of MT Summit XV*, page 117.

Durrani, N., Schmid, H., and Fraser, A. (2011). A joint sequence translation model with integrated reordering. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1045–1054, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dyer, C., Muresan, S., and Resnik, P. (2008). Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020. Association for Computational Linguistics.

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

Elghafari, A., Meurers, D., and Wunsch, H. (2010). Exploring the data-driven prediction of prepositions in english. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 267–275. Association for Computational Linguistics.

Ellis, R. (1994). *The study of second language acquisition*. Oxford University.

Federmann, C. (2010). Appraise: An open-source toolkit for manual phrase-based evaluation of translations. In *Proc. of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*. ELRA.

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*, pages 578–587, Denver, Colorado. Association for Computational Linguistics.

Felice, M. and Yuan, Z. (2014). Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–126, Gothenburg, Sweden. Association for Computational Linguistics.

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*, pages 15–24, Baltimore, Maryland. Association for Computational Linguistics.

Fossati, D. and Di Eugenio, B. (2007). A mixed trigrams approach for context sensitive spell checking. In *Computational Linguistics and Intelligent Text Processing*, pages 623–633. Springer.

Foster, J. and Andersen, Ø. E. (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*, pages 82–90. Association for Computational Linguistics.

Fraser, I. S. and Hodson, L. M. (1978). Twenty-one kicks at the grammar horse: Close-up: Grammar and composition. *English journal*, 67(9):49–54.

Gamon, M. (2010). Using mostly native data to correct errors in learners' writing: A meta-classifier approach. In *Proceedings of NAACL 2010*. Association for Computational Linguistics.

Gamon, M., Gao, J., Brockett, C., Klementiev, A., Dolan, W. B., Belenko, D., and Vanderwende, L. (2008). Using contextual speller techniques and language modeling for esl error correction. In *IJCNLP*, volume 8.

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*, pages 37–44. Association for Computational Linguistics.

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. *Calico Journal, Vol 26, No. 3.*

Gao, Q. and Vogel, S. (2008). Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57. ACL.

Golding, A. R. (1995). A bayesian hybrid method for context-sensitive spelling correction. In *In Proceedings of the Third Workshop on Very Large Corpora.*

Golding, A. R. and Roth, D. (1996). Applying winnow to context-sensitive spelling correction. In *Machine Learning, Proceedings of the Thirteenth International Conference (ICML '96)*, pages 182–190.

Golding, A. R. and Roth, D. (1999). A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

Golding, A. R. and Schabes, Y. (1996). Combining trigram-based and feature-based methods for context-sensitive spelling correction. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 71–78.

Grundkiewicz, R. (2013a). Automatic extraction of Polish language errors from text edition history. In *Text, Speech, and Dialogue — 16th International Conference, TSD 2013*, volume 8082 of *Lecture Notes in Computer Science*, pages 129–136, Plzen, Czech. Springer Berlin Heidelberg.

Grundkiewicz, R. (2013b). Errano: a tool for semi-automatic annotation of language errors. In *Proceedings of the 6th Language & Technology Conference*, pages 309–313, Poznan, Poland.

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 Przepiórkowski, A. and Ogrodniczuk, M., editors, *Advances in Natural Language Processing — Lecture Notes in Computer Science*, volume 8686, pages 478–490. Springer.

Grundkiewicz, R. and Junczys-Dowmunt, M. (2015). Grammatical error correction with (almost) no linguistic knowledge. In *Proceedings of the 7th Language & Technology Conference*, pages 240–245, Poznan, Poland.

Grundkiewicz, R. and Junczys-Dowmunt, M. (2017). Reinvestigating the classifiation approach to the article and preposition error correction. *Lecture Notes in Artificial Intelligence, Springer Verlag*, Human Language Technologies as a Challenge for Computer Science and Linguistics. To appear in the LTC 2015 post-conference volume.

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*, pages 461–470, Lisbon, Portugal. Association for Computational Linguistics.

Habash, N. (2008). Four techniques for online handling of out-of-vocabulary words in arabic-english statistical machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 57–60. Association for Computational Linguistics.

Han, N.-R., Chodorow, M., and Leacock, C. (2006). Detecting errors in english article usage by non-native speakers. *Natural Language Engineering*, 12(2):115–129.

Hasler, E., Haddow, B., and Koehn, P. (2012). Sparse lexicalised features and topic adaptation for smt. In *International Workshop on Spoken Language Translation (IWSLT) 2012*.

Hdez, 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*, pages 53–59, Baltimore, Maryland. Association for Computational Linguistics.

Heafield, K. and Lavie, A. (2010). Combining machine translation output with open source: The Carnegie Mellon multi-engine machine translation scheme. *The Prague Bulletin of Mathematical Linguistics*, 93:27–36.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. (2013). Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Heidorn, G. E. (2000). Intelligent Writing Assistance. In Dale, R., Moisl, H., and Somers, H., editors, *Handbook of Natural Language Processing*, pages 181–207. Dekker, New York, NY, USA.

Heidorn, G. E., Jensen, K., Miller, L. A., Byrd, R. J., and Chodorow, M. (1982). The epistle text-critiquing system. *IBM Systems Journal*, 21:305–326.

Herbrich, R., Minka, T., and Graepel, T. (2007). Trueskill(tm): A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press.

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)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Hinkle, D. E., Wiersma, W., and Jurs, S. G. (2003). Applied statistics for the behavioral sciences.

Hirst, G. and Budanitsky, A. (2005). Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(1):87.

Hoang, D. T., Chollampatt, S., and Ng, H. T. (2016). Exploiting n-best hypotheses to improve an smt approach to grammatical error correction. *arXiv preprint arXiv:1606.00210*.

Holan, T., Kubon, V., and Platek, M. (1997). A prototype of a grammar checker for czech. In *ANLP*, pages 147–154.

Hopkins, M. and May, J. (2011). Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1352–1362, Stroudsburg, USA. Association for Computational Linguistics.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T., and Isahara, H. (2003). Automatic error detection in the japanese learners' english spoken data. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 2*, pages 145–148. Association for Computational Linguistics.

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*, pages 74–81, Sofia, Bulgaria. Association for Computational Linguistics.

Junczys-Dowmunt, M. (2012). A phrase table without phrases: Rank encoding for better phrase table compression. In *16th Annual Conference of the European Association for Machine Translation (EAMT)*, pages 245–252, Trento, Italy.

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*, pages 25–33, Baltimore, Maryland. Association for Computational Linguistics.

Junczys-Dowmunt, M. and Grundkiewicz, R. (2016). Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Austin, Texas. Association for Computational Linguistics.

Koehn, P. (2010). *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Koehn, P. (2012). Simulating human judgment in machine translation evaluation campaigns. In *International Workshop on Spoken Language Translation*, pages 179–184.

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 *Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics.

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*, pages 48–54. Association for Computational Linguistics.

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Comput. Surv.*, pages 377–439.

Leacock, C., Chodorow, M., Gamon, M., and Tetreault, J. (2010). Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies*, 3(1):1–134.

Lee, J. (2004). Automatic article restoration. In *Proceedings of the Student Research Workshop at HLT-NAACL 2004*, HLT-SRWS '04, pages 31–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lee, J. and Seneff, S. (2008). Correcting misuse of verb forms. In *Proceedings of ACL-08: HLT*, pages 174–182, Columbus, Ohio. Association for Computational Linguistics.

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*, pages 65–73, Baltimore, Maryland. Association for Computational Linguistics.

Lee, L.-H., Yu, L.-C., Chang, L.-P., et al. (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*, pages 1–6.

Levenshtein, V. I. (1966). Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10.

Lewis, M. P., Simons, G. F., and Fennig, C. D. (2015). *Ethnologue: Languages of the world*, volume 18.

Liu, V. and Curran, J. R. (2006). Web text corpus for natural language processing. In *European Chapter of the Association for Computational Linguistics*. The Association for Computer Linguistics.

MacDonald, N. H., Frase, L. T., Gingrich, P. S., and Keenan, S. A. (1982). The Writer's Workbench: Computer aids for text analysis. *IEEE Trans. Communications*, COM-30(1):105–110.

Macháček, M. and Bojar, O. (2013). Results of the WMT13 metrics shared task. In *Proc. of the Eighth Workshop on Statistical Machine Translation*, pages 45–51. ACL.

Macháček, M. and Bojar, O. (2014a). Results of the WMT14 metrics shared task. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 293–301. ACL.

Macháček, M. and Bojar, O. (2014b). Results of the WMT14 metrics shared task. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 293–301. ACL.

Madnani, N., Tetreault, J., Chodorow, M., 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: short papers-Volume 2*, pages 508–513. Association for Computational Linguistics.

Maier, D. (1978). The Complexity of Some Problems on Subsequences and Supersequences. *J. ACM*, 25(2):322–336.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Markov, A. A. (1960). The theory of algorithms. *Am. Math. Soc. Transl.*, 15:1–14.

Max, A. and Wisniewski, G. (2010). Mining Naturally-occurring Corrections and Paraphrases from Wikipedia's Revision History. In *Proceedings of LREC*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Miłkowski, M. (2008). Automated Building of Error Corpora of Polish. In *Corpus Linguistics, Computer Tools, and Applications — State of the Art*, pages 631–639. Peter Lang.

Miłkowski, M. (2009). Automating rule generation for grammar checkers. In *Explorations across Languages and Corpora, PALC 2009*, pages 123–133.

Miłkowski, M. (2010). Developing an open-source, rule-based proofreading tool. *Software: Practice and Experience*, 40(7):543–566.

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*, pages 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 *The 5th International Joint Conference on Natural Language Processing*, pages 147–155.

Mizumoto, T. and Matsumoto, Y. (2016). Discriminative reranking for grammatical error correction with statistical machine translation. In *Proceedings of NAACL-HLT*, pages 1133–1138.

Mohit, B., Rozovskaya, A., Habash, N., Zaghouani, W., and Obeid, O. (2014a). The first qalb shared task on automatic text correction for arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47, Doha, Qatar. Association for Computational Linguistics.

Mohit, B., Rozovskaya, A., Habash, N., Zaghouani, W., and Obeid, O. (2014b). The first qalb shared task on automatic text correction for arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47, Doha, Qatar. Association for Computational Linguistics.

Moore, R. C. and Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 220–224, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mozgovoy, M. (2011). Dependency-based rules for grammar checking with languagetool. In Ganzha, M., Maciaszek, L. A., and Paprzycki, M., editors, *FedCSIS*, pages 209–212.

Naber, D. (2003). *A rule-based style and grammar checker*. PhD thesis, Bielefeld University Bielefeld, Germany.

Napoles, C., Sakaguchi, K., and Tetreault, J. (2016). There's no comparison: Reference-less evaluation metrics in grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2109–2115, Austin, Texas. Association for Computational Linguistics.

Napoles, C., Sakaguchi, K., and Tetreault, J. (2017). JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848.

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*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

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*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Nicholls, D. (2003). The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, USA. Association for Computational Linguistics.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proc. of the 40th Annual Meeting on ACL*, pages 311–318. ACL.

Park, J. C., Palmer, M. S., and Washburn, C. (1997). An english grammar checker as a writing aid for students of english as a second language. In *ANLP*, page 24.

Parker, R., Graff, D., Kong, J., Chen, K., and Maeda, K. (2011). English gigaword fifth edition, linguistic data consortium. Technical report, Technical Report. Linguistic Data Consortium, Philadelphia.

Perez-Cortes, J. C., Amengual, J.-C., Arlandis, J., and Llobet, R. (2000). Stochastic error-correcting parsing for ocr post-processing. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 405–408. IEEE.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

Roth, D. (1998). Learning to resolve natural language ambiguities: A unified approach. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 806–813, Menlo Park, CA, USA. American Association for Artificial Intelligence.

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*, pages 26–35, Beijing, China. Association for Computational Linguistics.

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*, pages 13–19, Sofia, Bulgaria. Association for Computational Linguistics.

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*, pages 34–42, Baltimore, Maryland. Association for Computational Linguistics.

Rozovskaya, A. and Roth, D. (2010a). Annotating ESL errors: Challenges and rewards.

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*, EMNLP '10, pages 961–970, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rozovskaya, A. and Roth, D. (2010c). Training paradigms for correcting errors in grammar and usage. In *North American Chapter of the Association for Computational Linguistics*.

Rozovskaya, A. and Roth, D. (2014). Building a state-of-the-art grammatical error correction system.

Rozovskaya, A. and Roth, D. (2016). Grammatical error correction: Machine translation and classifiers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2205–2215, Berlin, Germany. Association for Computational Linguistics.

Rozovskaya, A., Roth, D., and Srikumar, V. (2014b). Correcting grammatical verb errors. In *European Chapter of the Association for Computational Linguistics*.

Rozovskaya, A., Sammons, M., and Roth, D. (2012). The ui system in the hoo 2012 shared task on error correction.

Sakaguchi, K., Napoles, C., Post, M., and Tetreault, J. (2016). Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182.

Sakaguchi, K., Post, M., and Van Durme, B. (2014). Efficient elicitation of annotations for human evaluation of machine translation. In *Proc. of the Ninth Workshop on Statistical Machine Translation*, pages 1–11. ACL.

Sammut, C. and Webb, G. (2011). *Encyclopedia of machine learning*. Springer Science & Business Media.

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)*, pages 708–713, Sofia, Bulgaria. Association for Computational Linguistics.

Schneider, D. and McCoy, K. F. (1998). Recognizing syntactic errors in the writing of second language learners. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*, ACL '98, pages 1198–1204. Association for Computational Linguistics.

Selinker, L. and Gass, S. M. (1992). *Language transfer in language learning*. J. Benjamins Publishing Company.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Shannon, C. E. (1951). Prediction and entropy of printed english. *Bell Labs Technical Journal*, 30(1):50–64.

Stehouwer, H. and van Zaanen, M. (2009). Language models for contextual error detection and correction. In *Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference*, CLAGI '09, pages 41–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Susanto, R. H., Phandi, P., and Ng, H. T. (2014). System combination for grammatical error correction. pages 951–962.

Tajiri, T., Komachi, M., and Matsumoto, Y. (2012). Tense and aspect error correction for esl learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 198–202, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tamchyna, A., Fraser, A., Bojar, O., and Junczys-Dowmunt, M. (2016). Target-side context for discriminative models in statistical machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1704–1714, Berlin, Germany. Association for Computational Linguistics.

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*, ACLShort '10, pages 353–358, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tetreault, J. R. and Chodorow, M. (2008). The ups and downs of preposition error detection in esl writing. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 865–872, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tetreault, J. R. and Chodorow, M. (2009). Examining the use of region web counts for esl error detection. In *Web as Corpus Workshop (WAC5)*, page 71.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. 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, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Turner, J. and Charniak, E. (2007). Language modeling for determiner selection. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Vaswani, A., Zhao, Y., Fossum, V., and Chiang, D. (2013). Decoding with large-scale neural language models improves translation. In *Empirical Methods on Natural Language Processing*, pages 1387–1392. Association for Computational Linguistics.

Wagner, J., Foster, J., and van Genabith, J. (2007). A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors. In *The 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 112–121. Association for Computational Linguistics.

Wang, K., Thrasher, C., Viegas, E., Li, X., and Hsu, B.-j. P. (2010). An overview of microsoft web n-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, pages 45–48. Association for Computational Linguistics.

Wang, Y., Wang, L., Zeng, X., Wong, D. F., Chao, L. S., and Lu, Y. (2014). Factored statistical machine translation for grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 83–90, Baltimore, Maryland. Association for Computational Linguistics.

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*, pages 109–114, Sofia, Bulgaria. Association for Computational Linguistics.

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*, pages 91–95, Baltimore, Maryland. Association for Computational Linguistics.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y. (2016). Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

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-Volume 1*, pages 180–189. Association for Computational Linguistics.

Yi, X., Gao, J., and Dolan, W. B. (2008). A web-based english proofing system for english as a second language users. In *Third International Joint Conference on Natural Language Processing, IJCNLP 2008, Hyderabad, India, January 7-12, 2008*, pages 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*, pages 26–33, Sofia, Bulgaria. Association for Computational Linguistics.

Yu, L.-C., Lee, L.-H., and Chang, L.-P. (2014). Overview of grammatical error diagnosis for learning chinese as a foreign language. In *Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA'14), Nara, Japan*, pages 42–47.

Yuan, Z. and Briscoe, T. (2016). Grammatical error correction using neural machine translation. In *Proceedings of NAACL-HLT*, pages 380–386.

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*, pages 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*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.

Zesch, T. (2012). Measuring Contextual Fitness Using Error Contexts Extracted from the Wikipedia Revision History. In *Proceedings of EACL*, pages 529–538.

Zhao, Y. and Ishikawa, M. K. H. (2015). Improving chinese grammatical error correction using corpus augmentation and hierarchical phrase-based statistical machine translation. *ACL-IJCNLP 2015*.