A Statistical Approach to Grammatical Error Correction

Daniel Hermann Richard Dahlmeier

NATIONAL UNIVERSITY OF SINGAPORE

2013

A Statistical Approach to Grammatical Error Correction

Daniel Hermann Richard Dahlmeier

(Dipl.-Inform.), University of Karlsruhe

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

NUS GRADUATE SCHOOL FOR INTEGRATIVE SCIENCES AND ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2013

Declaration

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

[illegible]

Daniel Hermann Richard Dahlmeier

25 May 2013

Acknowledgment

A doctoral thesis is rarely a single, monolithic piece of work. Typically it is the report of an inquisitive journey with all its surprises and discoveries. At the end of the journey, it is time to acknowledge all those that have contributed to it.

First and foremost, I would like to thank my supervisor Prof Ng Hwee Tou. His graduate course at NUS first introduced me to the fascinating field of natural language processing. With his sharp analytical skills and his almost uncanny accurateness and precision, Prof Ng has always been the most careful examiner of my work. If I could convince him of my ideas, I was certain that I could convince the audience at the next conference session as well. Discussions with him have been invaluable for me in sharpen my scientific skills.

Next, I would like to thank the other members of my thesis advisory committee, Prof Tan Chew Lim and Prof Lee Wee Sun. Their guidance and feedback during the time of my candidature has always been helpful and encouraging.

I would like to thank my friends at the NUS Graduate School for Integrative Sciences and Engineering and the School of Computing for support, helpful discussions, and fellowship.

Finally, I would like to thank my wife Yee Lin for her invaluable moral support throughout my graduate school years.

Contents

Abstract

A large part of the world's population regularly needs to communicate in English, even though English is not their native language. The goal of *automatic grammatical error correction* is to build computer programs that can provide automatic feedback about erroneous word usage and ill-formed grammatical constructions to a language learner. Grammatical error correction involves various aspects of computational linguistics, which makes the task an interesting research topic. At the same time, grammatical error correction has great potential for practical applications for language learners.

In this Ph.D. thesis, we pursue a statistical approach to grammatical error correction based on machine learning methods that advance the field in several directions. First, the NUS Corpus of Learner English, a one-million-word corpus of annotated learner English was created as part of this thesis. Based on this data set, we present a novel method that allows for training statistical classifiers with both learner and non-learner data and successfully apply it to article and preposition errors. Next, we focus on lexical choice errors and show that they are often caused by words with similar translations in the native language of the writer. We show that paraphrases induced through the native language of the writer can be exploited to automatically correct such errors. Fourth, we present a pipeline architecture that combines individual correction modules into an end-to-end correction system with state-of-the-art results. Finally, we present a novel beam-search decoder for grammatical error correction that can correct sentences which contain multiple and interacting errors. The decoder further improves over the state-of-the-art pipeline architecture, setting a new state of the art in grammatical error correction.

List of Tables

List of Figures

Chapter 1

Introduction

In an increasingly globalized world, it has become a necessity for everyone to learn one or more foreign languages. For anyone who is not from an English-speaking country, this necessarily includes English which has become the *lingua franca* for people around the world to communicate with one another if they do not speak the same language. English is not only spoken in countries with a native English-speaking population but is a global communication medium. The British Council estimated that in the year 2000, there were about one billion people learning English in the world. This number is expected to further increase to around two billion (Graddol, 2006). This means that soon around one third of the world's population will be learning English and that speakers of English as a foreign language (EFL) will greatly outnumber English native speakers. A large percentage of these non-native English speakers will be coming from Asia.

However, learning a foreign language is difficult. It requires years of continuous practice and corrective feedback from a proficient teacher. But even the most dedicated teacher cannot attend to her students 24 hours a day, and many students in developing countries do not have access to high quality language education at all. With the ubiquitous presence of modern computers and their increasing role in teaching and education, it seems attractive to utilize computers to help language learning students by providing corrective feedback on grammatical errors in an automatic fashion. To accomplish this task, the computer would have to be equipped with a set of rules that describe how to correct the language learner. But language is extremely complex and constantly evolving. It is very difficult to explicitly write down the exact rules that define a grammatical sentence. Manually engineered rules therefore cannot cover all the variety that is observed in real language data. To make matters worse, every rule has its own exceptions, as anyone who has studied a foreign language can attest.

The success of statistical approaches to natural language processing (NLP) offers a different solution. Instead of trying to define all the rules of a language and then implement these rules in a computer algorithm, the statistical approach to natural language processing lets a learning algorithm learn the rules from data. The "statistical revolution" that has taken place over the last two decades has resulted in great progress in many areas of natural language processing. The goal of this thesis is to bring some of this progress to grammatical error correction. To see why computers can at least potentially succeed in learning a language, let us take a look at the following comparison by Philipp Koehn (2006) to see how much exposure to a language a human can actually get during a lifetime of studying and how much text data can be processed by computer algorithms. The comparison was made in the context of statistical machine translation (SMT) but applies to language learning as well. If we assume that a human can read 10,000 words a day, and she studies every day without interruption, she can read about 3.5 million words a year, and about 300 million words during her lifetime. If we compare this number to the amount of text that is available to computers in electronic form today, 300 million words appear quite humble. Large text corpora used in natural language processing already contain a few billion words and the World Wide Web is estimated to contain over a trillion words. Thus, computers have access to much more text than any human can read in a lifetime. Thus, the computer could at least in principle be able to "learn" a language just by seeing millions and millions of examples. While we focus solely on English in this thesis, the methods described in this thesis have applicability to other languages as well.

1.1 The Goal of Grammatical Error Correction

So what specifically is the goal of automatic grammatical error correction? Casually speaking, the goal of grammatical error correction is to build a machine which takes as input text written by a language learner, analyzes the text to detect and correct any grammatical errors, and outputs a corrected, fluent version of the input, possibly together with some explanation or analysis. As such, the task of grammatical error correction can be thought of as "decoding" the learner input text to recover the text that the learner wanted to express but was unable to construct properly. Grammatical error correction involves various aspects of computational linguistics, like language modeling, syntax, and semantics, which makes the task interesting and at the same time challenging from a research perspective. At the same time, grammatical error correction has great potential for practical applications, such as authoring aids and educational software language learning and assessment.

1.2 Contributions of this Thesis

Although considerable progress has been made in grammatical error correction, research has been hampered by a number of obstacles. In this section, we describe the contributions of this thesis to overcome some of these obstacles.

1.2.1 Creating a Large Annotated Learner Corpus

Statistical methods require data. The data is used to train statistical models and to evaluate the models' predictions with respect to the human annotation on a held-out test set. For most natural language processing tasks, the community has already created annotated data sets, e.g., the Penn Treebank corpus (Marcus et al., 1993) for part of speech tagging and parsing, or the data sets of the Workshop for Machine Translation (Callison-Burch et al., 2012). Despite the growing interest in grammatical error correction, there has been no large annotated learner corpus available for research in grammatical error correction until recently. The existing annotated learner corpora were all either too small or proprietary and not available to the research community.

The first contribution of this thesis is the creation of the *NUS Corpus of Learner English (NUCLE)*. NUCLE consists of about 1,400 essays written by EFL university students on a wide range of topics. It contains over one million words which are completely annotated with error tags and corrections. All annotations have been performed by professional English instructors. The details of the corpus are described in Chapter 3. NUCLE is the currently the largest annotated learner corpus that is freely available to the community for research purposes.

1.2.2 Evaluation of Grammatical Error Correction

Research in natural language processing is driven by empirical evaluation of the algorithms with regard to some metric of performance. The evaluation of grammatical error correction is done by measuring the similarity between the corrections proposed by a computer algorithm and a set of corrections proposed by a human expert. Unfortunately, evaluation is complicated by the fact that different sets of corrections can result in the same corrected sentence. In Chapter 3, we present a novel method for grammatical error correction that takes the ambiguity of the corrections into account. We show that this method solves problems in existing evaluation tools.

1.2.3 Learning Classifiers for Error Correction

As a result of the lack of learner data, the standard approach to grammatical error correction has been to train an off-the-shelf classifier to re-predict words in non-learner text based on the surrounding context. Training classifiers on non-learner text does not provide the same information that is found in annotated learner text. In particular, the information on which words are typically confused with which other words cannot be learned from the non-learner text as the data is assumed to be free of grammatical errors. Learning classifiers directly from annotated learner corpora is not well explored, as are methods that combine learner and non-learner text. In Chapter 4, we present a novel approach to grammatical error correction based on Alternating Structure Optimization (ASO) (Ando and Zhang, 2005). The approach is able to train models on annotated learner corpora while still taking advantage of large non-learner corpora. We evaluate our proposed ASO method on the task of article and preposition error correction. Our experiments show that the proposed ASO algorithm significantly improves over two baselines trained on non-learner text and learner text, respectively. It also outperforms two commercial grammar checking software packages in a manual evaluation.

1.2.4 Lexical Choice Error Correction with Paraphrases

Virtually all existing approaches to grammatical error correction assume a fixed *confusion set* of possible correction choices that is known beforehand. This works fine for error categories like articles and prepositions, but for more general errors that involve wrong word choices of nouns and verbs, it is much more difficult to define a suitable confusion set. In Chapter 5, we present a novel approach for automatic correction of lexical choice errors. The key observation is that words are potentially confusable for an EFL student if they have similar translations in the writer's native language, or in other words if they have the same semantics in the native language of the writer. While these types of transfer errors have been known in the EFL teaching literature, research in grammatical error correction has mostly ignored this fact. In Chapter 5, we empirically confirm that many lexical choice errors in the NUCLE corpus can be traced to similar translations in the writer's native language. Based on this result, we propose a novel approach for automatic lexical choice error correction. The key component in our approach is paraphrases which are automatically extracted from a parallel corpus of English and the writer's native language. The proposed approach outperforms traditional approaches based on edit distance, homophones, and WordNet synonyms on a test set of real-world learner data in an automatic and a human evaluation.

1.2.5 A Pipeline Architecture for Error Correction

Research in grammatical error correction has typically concentrated on a single error category in isolation. To build practical error correction applications, the components for different error categories need to be combined into an end-to-end error correction system. In Chapter 6, we present a general architecture for error correction that combines separate correction steps into a pipeline of correction steps. The architecture is evaluated in the context of two shared tasks and achieves state-of-the-art results.

1.2.6 A Beam-Search Decoder for Grammatical Error Correction

Although the pipeline approach to error correction achieves state-of-the-art results, it suffers from some serious shortcomings. Each classifier corrects a single word for a specific error category individually. This ignores dependencies between the words in a sentence. Also, by conditioning on the surrounding context, the classifier implicitly assumes that the surrounding context is free of grammatical errors, which is often not the case. Finally, the classifier typically has to commit to a single one-best prediction and is not able to change its decision later or explore multiple corrections. Instead of correcting each word individually, we would like to perform global inference over corrections of whole sentences which can contain multiple and interacting errors. In Chapter 7, we present a novel beam-search decoder for grammatical error correction that extends the classification approach to a more general decoder framework similar to the approaches common in statistical machine translation. The decoder performs an iterative search over possible sentence-level hypotheses to find the best correct sentence for the input sentence. We evaluate the decoder in the context of two shared tasks on grammatical error correction and show that the decoder improves upon a state-of-the-art pipeline model in both cases.

1.3 Summary of Contributions

In summary, the contributions of this thesis are as follows. First, the NUCLE learner corpus, a fully annotated, one-million word corpus of learner English was created. Second, we present an improved evaluation method for grammatical error correction.

Third, we develop a novel method to train statistical classifiers for error correction based on alternating structure optimization. Fourth, we empirically show that lexical choice errors are often linked to similar translations in the learner's native language and that paraphrases induced through the native language can be used to correct these errors. Fifth, we present a pipeline architecture for error correction systems with state-of-the-art results. Sixth, we develop a novel beam-search decoder for grammatical error correction that improves over the existing state-of-the-art results.

1.4 Organization of the Thesis

The remainder of this thesis is organized as follows. The next chapter gives an overview of related work in grammatical error correction. Chapter 3 describes the NUCLE corpus and other data sets for grammatical error correction, and evaluation metrics. Chapter 4 describes the alternating structure optimization algorithm for error correction. Chapter 5 describes the lexical choice error correction work based on paraphrases. Chapter 6 describes the pipeline architecture for grammatical error correction. Chapter 7 describes the beam-search decoder. Chapter 8 concludes the thesis.

Chapter 2

Related Work

Research can never be done in a vacuum in isolation from the body of existing knowledge that has already been accumulated. Instead, every new scientific result has to be presented in the context of the work that precedes it. A thorough review of related work is therefore part of any serious scientific endeavor. “Standing on the shoulders of giants”, as Newton famously put it, allows us to see further than we could have otherwise. The review also helps researchers to understand the problem at hand and serves as the starting point for finding improvements of and alternatives to existing approaches. For empirical disciplines like natural language processing, previously published methods serve as a baseline to quantify the improvement of the presented methods. Finally, the study of existing work should give credit to the academic community where credit is due.

From its beginning, research in grammatical error correction has been closely linked to the development of grammar checking tools for text processing. While the earliest tools such as the Unix Writer’s Workbench (MacDonald et al., 1982) were based purely on string matching algorithms, later systems, such as IBM’s Epistle (Heidorn et al., 1982), already started using some form of linguistic analysis. The correction mechanisms of these early systems were based on logical re-write rules which were engineered by human experts. The Microsoft NLP analysis system that underlies the grammar checking functionality of Microsoft Word is based on such a rule-based framework

(Heidorn, 2000).

With the availability of large-scale computational grammars, parser-based methods for detecting and correcting grammatical errors emerged (Heift and Schulze, 2007). Early parser-based approaches to grammatical error correction tried to devise parsing algorithms that are robust enough to parse learner text with grammatical errors and at the same time provide sufficient information for correcting the grammatical errors. Robust parsing of text with grammatical errors can be achieved through different strategies, for example by introducing special "mal-rules" to parse grammatically ill-formed constructions (Schneider and McCoy, 1998) or by relaxing parse constraints (Hagen, 1995; Schwind, 1990). More recent work has tried to leverage statistical parsers learned from syntactically annotated treebanks with automatically introduced errors (Foster, 2007).

Because early work in grammatical error correction was primarily based on manually engineered rules, it was not able to cover the full variety of grammatical errors that are made by language learners. The advent of statistical NLP brought about a set of new methods that could make predictions about words in context based on previously observed training data. These algorithms were applicable to a wide range of tasks, such as word sense disambiguation (WSD) (Gale et al., 1992; Ng and Lee, 1996; Lee and Ng, 2002), accent restoration (Yarowsky, 1994), context-sensitive spelling error correction (Golding, 1995), and error correction (Knight and Chander, 1994).

In the remainder of this chapter, we give a more detailed overview about related statistical work on grammatical error correction. In particular, we focus on article errors, preposition errors, lexical choice errors, and decoding-based methods in error correction. We also highlight the differences to our work presented in this thesis. A more comprehensive survey of grammatical error correction for language learners can be found in the excellent book by Claudia Leacock *et al.* (2010).

2.1 Article Errors

The seminal work on automatic grammatical error correction was done by Knight and Chander (1994). They were motivated by the idea of automatic post-editing for low quality English texts produced by either computers, e.g., machine translation systems, or language learners. As a first step towards the goal of a general post-editor system, they presented a system that automatically predicts which of the three articles *a, an* or *the* should be used for an English noun phrase in a given context. The system used a decision tree classifier trained on English noun phrase examples from the Wall Street Journal. Each noun phrase is one training example. The noun phrase and its context are represented by a set of binary feature functions, e.g., surrounding words, head word of the noun phrase, part of speech tags, and the article used by the writer is the class label. The idea to train a classifier to predict the correct English word given some feature representation of the surrounding context has had a major influence on grammatical error correction. Subsequent work on article corrections has changed the set of articles to the indefinite article *a*, the definite article *the*, and the null article ϵ (meaning that the noun phrase does not have an article). This confusion set covers article insertion, deletion, and replacement errors. The distinction between the indefinite articles *a* and *an* can easily be done with a set of rules in a post-processing step. Most work on article correction has stayed with the classification approach and has been concerned with designing better features and testing different classifiers, including memory-based learning (Minnen et al., 2000), decision tree learning (Nagata et al., 2006), and logistic regression (Lee, 2004; Han et al., 2006; De Felice, 2008). Gamon *et al.* (2008) divided the three-way classification task into a binary presence vs. absence classification step followed by a binary definite vs. indefinite classification. They also added an additional language model filter step after the classification. Any proposed correction that received a lower language model score than the original sentence was discarded.

All of the above works only use non-learner text for training. A shortcoming of training on non-learner text is that the it assumes that all confusions between the articles are equally likely. That assumption does not hold true in practice where some confusions happen more frequently than others. Most importantly, the correct article is in most cases the same as the article used by the writer, as grammatical errors are typically sparse and most articles in a given learner text are correct. Therefore the observed article used by the writer is an important feature. This observation was first made by Rozovskaya and Roth (2010b). However, to train a classifier that uses the "observed article" feature, it is necessary to have annotated learner data that contains both the article chosen by the writer and the correct article chosen by a human expert, e.g., an English teacher. This type of data is much more difficult to obtain than normal non-learner corpora. Rozovskaya and Roth produced error annotations for a subset of the International Corpus of Learner English (ICLE) (Granger et al., 2002) but the data set was too small to directly utilize it to train classifiers. Instead, Rozovskaya and Roth chose a different strategy. They used the learner corpus to derive frequency statistics of learner errors and then introduced artificial errors in a larger non-learner corpus with a frequency similar to the observed frequency in learner text. While this injects some information from the learner corpus into the training process, their method for introducing artificial errors in learner text does not take into account the context of the article. In practice, the context of an article has an effect on how likely a learner will confuse two articles. For example, the article choice before pronouns or proper nouns is much easier to learn than other more ambiguous contexts, like *I am going on {a, the, ϵ} holiday*, where even native speakers might have to carefully consider the context before making a decision. In the end, introducing artificial learner errors in native text in a way that closely imitates learners' behavior is just as difficult as correcting errors in learner text. Artificially created learner errors might not represent the true distribution of learner errors accurately.

There have been few approaches to learn classifiers directly from learner corpora. Izumi *et al.* (2003) worked on automatic error correction for spoken texts from Japanese learners. The learner data that they had available was too small to learn reliable classifiers for most error categories but they presented some results for article corrections. They also explored adding additional corrected sentences or sentences with artificial errors to the training data to address the data sparsity problem. This shows again the need for a large annotated learner corpus. Almost no work has investigated ways to combine learner and non-learner text for training. The only exception is Gamon (2010), who combined features from the output of logistic-regression classifiers and language models trained on non-learner text in a meta-classifier trained on learner text.

Finally, researchers have investigated article correction in connection with web-based models in NLP (Lapata and Keller, 2005; Yi et al., 2008). These methods do not use classifiers, but rely on simple N-gram counts or page hits from the Web.

We see that article error correction is still largely treated as a generic classification problem of predicting the correct article for a noun phrase. Non-learner text has been the main source of training data because it is cheap and readily available in large quantities and because it is easy to create "fill-in-the-blanks" training examples by using the original article as the class label without the need to perform any manual annotation. A classifier is then trained to re-predict the original article based on the context. At the same time, there has been work that suggests that learner text is a more valuable resource for training. Especially the original article used by the writer is a very valuable feature because article errors are sparse and the correct article is in many cases the same as the original article. In Chapter 4 of this thesis, we present an ASO learning algorithm for grammatical error correction. The algorithm has the advantage that it can make use of both the large amounts of non-learner text and the highly valuable, although limited, learner text.

2.2 Preposition Errors

Work on preposition errors has followed the same classification approach that was presented for article errors above. One difference from article correction is that preposition error correction has mainly focused on replacement errors of prepositions where the preposition written by the author needs to be replaced with another preposition, and less on preposition insertion and deletion errors. The set of prepositions that are considered for correction is fixed to a list of frequent English prepositions, typically between 10 and 36. The prepositions are the possible class labels for the classifier and the surrounding context of a preposition provides the features. The features for preposition errors, of course, differ from the features for articles. The work by Chodorow *et. al* (2007) is one of the first examples of a classifier-based approach to preposition correction. They use a maximum entropy classifier and features from surrounding words, part of speech tags, and chunks. Subsequent work aimed to improve their approach (Tetreault and Chodorow, 2008b; Tetreault and Chodorow, 2008a), for example through the inclusion of features from syntactic parse trees (Lee and Knutsson, 2008; De Felice, 2008; Tetreault et al., 2010).

Features in natural language classification tasks are usually binary valued and signify the presence or absence of a particular contextual predicate, for example, the presence or absence of a particular N-gram. An alternative type of features is web-scale N-gram features that were proposed by Bergsma *et al.* (2009) for a number of natural language processing tasks. In contrast to binary features, web-scale N-gram features consist of log-counts of N-grams in a web-scale corpus and take real values. By replacing the target word in the center of the N-gram windows with different possible choice, e.g., different prepositions, the counts for different target words can be computed. The log-counts can be used as features in a standard supervised learning algorithm. Bergsma *et al.* showed that web-scale N-gram features are very effective for predicting prepositions in non-learner text but they did not evaluate their method on real examples of learner texts.

All of the above works have focused on preposition replacement errors. Gamon *et al.* (2008) is one of the few approaches that considered preposition insertion, deletion, and replacement errors. Using the same approach as presented for articles, they divided the task into a binary presence-absence classification step, followed by a multi-class preposition selection step, and a language model filter.

All of the above works only use non-learner text for training. This assumes that a preposition is equally confusable with every other prepositions which is not true. The information on how likely one preposition is confused with another preposition is a piece of important information that is not available when training classifiers on non-learner texts. Han *et al.* (2010) showed that training a preposition correction classifier on annotated learner texts gives better performance than training on non-learner texts. For their experiments, they used the Chungdahm English Learner Corpus, a corpus of essays written by Korean students in language schools run by Chungdahm Learning Inc. Although the Chungdahm corpus is very large (> 130 million words), it is only partially annotated and is not available for research purposes because of its proprietary nature. Rozovskaya and Roth (2010a) explore different strategies for injecting knowledge about the fact that a preposition is not equally confusable with all other prepositions, including restricting the confusion set to different subsets of prepositions and the generation of artificial learner errors in non-learner texts based on statistics from learner corpora. Almost no work has investigated ways to combine learner and non-learner text for training, with the notable exception of the meta-classification approach from (Gamon, 2010) that combined features from the output of logistic-regression classifiers and language models.

As we have already observed in the case of articles, preposition correction has largely been treated as a generic classification problem of re-predicting the original preposition in non-learner text. In the case of prepositions, the confusion set of possible choices for the classifier is several times larger than in the case of articles. That makes the task considerably harder than the article correction task. It is therefore not surprising that preposition correction typically requires more training data to achieve comparable classification performance. Our ASO algorithm presented in Chapter 4 has the advantage of being able to use large amounts of non-learner text which is particularly useful for the preposition correction task.

2.3 Lexical Choice Errors

Lexical choice errors have attracted comparatively less attention than article and preposition errors. The first problem when correcting lexical choice errors is that there is not a fixed confusion set of candidates to choose from. One direction of research that is concerned with lexical choice errors is collocation error correction. Collocations are sequences of words that are conventionally used together in a particular way. Previous work in collocation correction has relied on dictionaries or manually created databases to generate collocation candidates (Shei and Pain, 2000; Wible et al., 2003; Futagi et al., 2008). Other work has focused on finding candidates that collocate with similar words, e.g., verbs that appear with the same noun objects form a confusion set (Liu et al., 2009; Wu et al., 2010). The work presented by Chang *et al.* (2008) uses translation information to generate collocation candidates. That is similar to the approach presented in this thesis. However, they do not use automatically derived paraphrases from parallel corpora but bilingual dictionaries. Dictionaries usually have lower coverage, do not contain longer phrases or inflected forms, and do not provide any translation probability estimates. Also, their work focuses solely on verb-noun collocation errors, while the system presented in this thesis targets errors of arbitrary syntactic type.

Another direction on lexical choice correction is context-sensitive spelling error correction. Context-sensitive spelling error correction is the task of correcting spelling mistakes that result in another valid word, see for example (Golding and Roth, 1999). It has traditionally focused on a small number of pre-defined confusion sets, like homophones or frequent spelling errors. Even when the confusion sets were formed automatically, the similarity of words in a confusion set has been based on edit distance or similar phonetics (Carlson et al., 2001). In contrast, in this thesis, we focus on lexical choice errors that are related to similar semantics of the confused words instead of similar spelling or pronunciation.

Synonym extraction (Wu and Zhou, 2003), lexical substitution (McCarthy and Navigli, 2007) and paraphrasing (Madnani and Dorr, 2010) are related to lexical choice correction in the sense that they try to find semantically equivalent words or phrases.

However, there is a subtle but important difference between these tasks and lexical choice correction. In the former, the main criterion is whether the original phrase and the synonym or paraphrase candidate are substitutable, i.e., both form a grammatical sentence when substituted for each other in a particular context. In contrast, in lexical choice correction, the primarily interest is to find candidates which are *not substitutable* in their English context but *appear to be substitutable* in the native language of the writer, i.e., one forms a grammatical English sentence but the other does not.

Lexical choice errors cover a much broader range of words and parts of speech than closed set error classes like article and preposition errors. As a result, lexical choice errors cannot easily be cast as a generic classification problem. The difficulty of applying standard classification methods might be the reason why lexical choice errors have not received more attention in grammatical error correction yet. In Chapter 5, we show an easy and intuitive method to derive the confusion set of a word based on its translations in the native language of the writer. We further present an automatic method for correcting lexical choice errors with the help of paraphrases induced through the native language of the writer.

2.4 Decoding Approaches

The approaches that we have described so far can all be considered as part of the classifier-based approach to error correction. Alternatively, error correction can be viewed as a decoding problem that tries to "decode" the ungrammatical learner sentence to find the grammatically correct sentence, similar to statistical machine translation (Koehn, 2010). This approach is more general and can correct whole sentences with multiple and different errors. However, the decoding approach to error corrections has received little attention. Brockett *et al.* (2006) used a statistical machine translation system to correct errors involving mass noun errors. Because no large annotated learner corpus was available, the training data was created artificially from non-learner text. Lee and Seneff (2006) described a lattice-based correction system with a domain-specific grammar for spoken utterances from the flight domain. The work in (Désilets and Hermet, 2009) used simple round-trip translation with a standard SMT system to correct grammatical errors. Park and Levy (2011) proposed a noisy channel model for error correction. Their motivation to correct whole sentences is similar to the motivation that lead to the decoder presented in this thesis. But Park and Levy's proposed generative method differs substantially from the discriminative decoder proposed in this thesis. Their model does not allow the use of discriminative expert classifiers as our decoder does, but instead relies on a bigram language model to find grammatical corrections. Indeed, the authors point out that the language model often fails to distinguish grammatical and ungrammatical sentences.

In Chapter 7, we present a beam-search decoder framework that combines the strength of existing classification approaches with a search-based decoding approach. The idea that grammatical error correction should be seen as a sentence-level decoding task rather than a word-by-word classification task is a novel contribution of this thesis. Although some researchers have started to think in this direction, they have used existing generic decoding frameworks like SMT decoding and lattice decoding to solve the problem. While this sidesteps the non-trivial task of having to implement a decoding algorithm from scratch, the generic models are not able to incorporate task-specific models, such as existing classifier models for grammatical error correction. Our decoder model goes beyond simple classification and proposes a new, general framework for grammatical error correction. We see the beam-search decoder model as the most significant single contribution of this thesis.

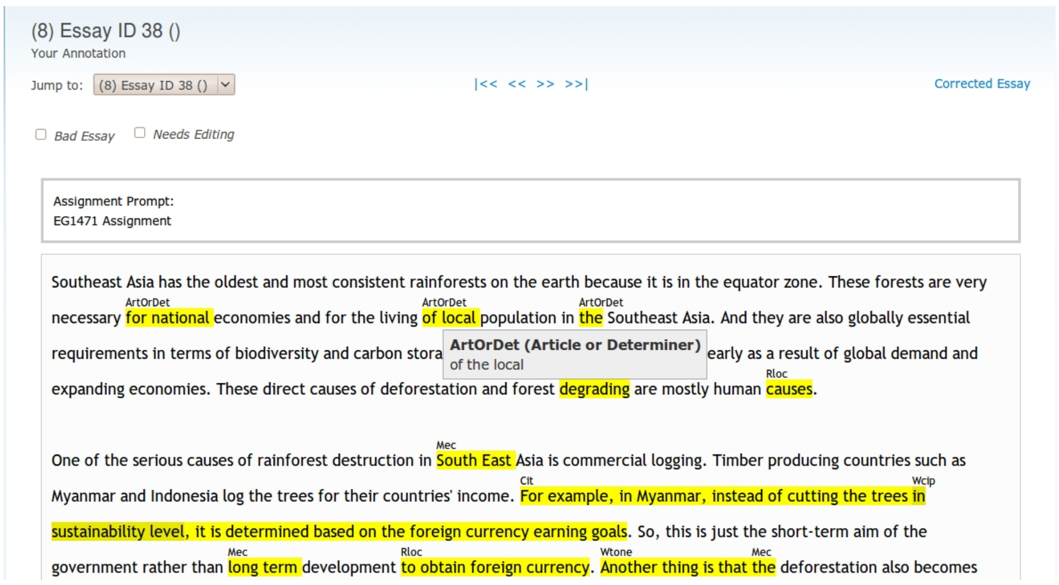
Chapter 3

Data Sets and Evaluation

In this chapter, we describe text corpora and evaluation measures for grammatical error correction. Most importantly, we introduce the NUS Corpus of Learner English that was created as part of this thesis. We also describe the data set of the Helping Our Own (HOO) shared tasks. For evaluation, we describe the standard measures of precision, recall, and F_1 score and a novel method, called *MaxMatch* (M^2), for computing these scores for grammatical error correction.

3.1 NUS Corpus of Learner English

The biggest obstacle that has held back research in grammatical error correction until recently has been the lack of a large annotated corpus of learner text that could serve as a standard resource for empirical approaches to grammatical error correction (Leacock et al., 2010). That is why we decided to create the first large, annotated corpus of learner texts that is available for research purposes: the NUS Corpus of Learner English (NUCLE). The corpus was built in collaboration with the NUS Center for English Language Communication (CELC). NUCLE consists of more than 1,400 student essays from undergraduate students at NUS with over one million words which are completely annotated with error tags and corrections. All annotations and corrections have been performed by professional English instructors. To the best of our knowledge, NUCLE is the first corpus of this size and quality that is available for research purposes. In this


(8) Essay ID 38 ()

Your Annotation

Jump to: (8) Essay ID 38 ()

Corrected Essay

Bad Essay Needs Editing

Assignment Prompt:
EG1471 Assignment

Southeast Asia has the oldest and most consistent rainforests on the earth because it is in the equator zone. These forests are very necessary for national economies and for the living of local population in the Southeast Asia. And they are also globally essential requirements in terms of biodiversity and carbon stora early as a result of global demand and expanding economies. These direct causes of deforestation and forest degrading are mostly human causes.

ArtOrDet (Article or Determiner)
of the local

One of the serious causes of rainforest destruction in South East Asia is commercial logging. Timber producing countries such as Myanmar and Indonesia log the trees for their countries' income. For example, in Myanmar, instead of cutting the trees in sustainability level, it is determined based on the foreign currency earning goals. So, this is just the short-term aim of the government rather than long term development to obtain foreign currency. Another thing is that the deforestation also becomes

Figure 3.1: The WAMP annotation interface

section, we describe the corpus in more detail.

3.1.1 Annotation Schema

Before starting the corpus creation, we had to develop a set of annotation guidelines. This was done in a pilot study between May and July 2009 in which three instructors from CELC participated. The instructors annotated a small set of student essays that had been collected by CELC. The annotation was performed using the Writing, Annotation, and Marking Platform (WAMP), an online annotation tool that was developed by the NUS NLP group specially for creating the NUCLE corpus. WAMP allows the annotators to work over the Internet using a web browser. Figure 3.1 shows a screen shot of the WAMP interface. Annotators can browse through a batch of essays that has been assigned to them and perform the following tasks:

- **Select** arbitrary, contiguous text spans using the cursor to identify grammatical errors.
- **Classify** errors by choosing an error tag from a drop-down menu.
- **Correct** errors by typing the correction into a text box.
- **Comment** to give additional explanations if necessary.

We wanted to impose as few constraints as possible on the annotators. Therefore, WAMP allows annotators to select arbitrary text spans, including overlapping text spans.

After some annotation trials, we decided to use a tag set which had been developed by CELC in a previous study. Some minor modifications were made to the original tag set based on the feedback of the annotators. The result of the pilot study was a tag set of error categories and an annotation guide that described how errors should be annotated. The tag set consists of 27 error categories which are listed in Table 3.1. It is important to note that our annotation schema does not only label each grammatical error with an error category, but it requires the annotator to provide a suitable correction for the error as well. The annotators were asked to provide a correction that would fix the grammatical error if the annotated word or phrase is replaced with the correction.

3.1.2 Annotator Agreement

How reliably can human annotators agree on whether a word or sentence is grammatically correct? The pilot annotation project gave us the opportunity to investigate this question in a quantitative analysis. Annotator agreement is also a common measure for how "difficult" a task is and servers as a test whether humans can reliable perform the annotation task with the given tag set. During the pilot study, we randomly sampled 100 essays for measuring annotator agreement. The essays were then annotated by our three annotators in a way that each essay was annotated independently by two annotators. Four essays had to be discarded as they were of very poor quality and did not allow for any meaningful correction. This left us with 96 essays with double annotation.

Comparing two sets of annotation is complicated by the fact that the set of annotations that corrects an input text to a corrected output text is ambiguous (see Section 3.4 below for details). In other words, it is possible that two different sets of annotations produce the same correction. For example, one annotator could choose to select a whole phrase as one error, while the other annotator selects each word individually. Our annotation guidelines asked annotators to select the minimum span that is necessary to correct the error, but we do not enforce any hard constraints and different annotators

Error Tag	Error Category	Description / Example
Vt	Verb Tense	A university **[had conducted | conducted]** the survey last year.
Vm	Verb modal	No one **[will | would]** bother to consider a natural balance.
V0	Missing verb	This **[may | may be]** due to a traditional notion that boys would be the main labor force in a farm family.
Vform	Verb form	Will the child blame the parents after he **[growing | grows]** up?
SVA	Subject-verb-agreement	The boy **[play | plays]** soccer.
ArtOrDet	Article or Determiner	From the ethical aspect, sex selection technology should not be used in **[non-medical | a non-medical]** situation.
Nn	Noun Number	Sex selection should therefore be used for medical **[reason | reasons]** and nothing else.
Npos	Noun possessive	The education of **[mother's | mothers]** is a significant factor in reducing son preference.
Pform	Pronoun form	90% of couples seek treatment for family balancing reasons and 80% of **[those | them]** want girls.
Pref	Pronoun reference	Moreover, children may find it hard to communicate with **[his/her | their]** parents.
Wcip	Wrong collocation/idiom/preposition	Singapore, for example, has invested heavily **[on | in]** the establishment of Biopolis
Wa	Acronyms	Using acronyms without explaining what they stand for.
Wform	Word form	Sex-selection may also result in **[addition | additional]** stress for the family.
Wtone	Tone	**[Isn't it | Is it not]** what you always dreamed for?
Srun	Runons, comma splice	**[Do spare some thought and time, we can make a difference! | Do spare some thought and time. We can make a difference!]** (Should be split into two sentences)

Table 3.1: NUCLE error categories. Grammatical errors in the example are printed in bold face in the form **[<mistake> | <correction>]**.

Error Tag	Error Category	Description / Example
Smod	Dangling modifier	Knowing the pitfalls ahead, **[the issue the administration has to manage. | the issue the administration has to manage cannot be underestimated.]** (Possible completion of sentence)
Spar	Parallelism	The use of sex selection would prevent rather than **[contributing | contribute]** to a distorted sex ratio.
Sfrag	Fragment	Although he is a student from the Arts faculty.
Ssub	Subordinate clause	It is the wrong mindset of people that boys are more superior than girls **[should | that should]** be corrected.
WOinc	Incorrect sentence form	Why can **[not we | we not]** choose more intelligent and beautiful babies?
WOadv	Adverb/adjective position	It is similar to the murder of many valuable lives **[only based | based only]** on the couple's own wish.
Trans	Link words/phrases	In the process of selecting the gender of the child, ethical problems arise **[where | because]** many innocent lives of unborn fetuses are taken away.
Mec	Punctuation, capitalization, spelling, typos	The **[affect | effect]** of that policy has yet to be felt.
Rloc	Local redundancy	Currently, abortion is available to end a life only **[because of | because]** the fetus or embryo has the wrong sex.
Cit	Citation	Poor citation practice.
Others	Other errors	Any error that does not fit into any other category, but can still be corrected.
Um	Unclear meaning	The quality of the passage is so poor that it cannot be corrected.

Table 3.1: (continued)

can have a different perception of where an error starts or ends.

An especially difficult case is the annotation of omission errors, for example missing articles. Selecting a range of whitespace characters is difficult for annotators, especially if the annotation tool is web-based as whitespace is variable in web pages. We asked annotators to select the previous and/or next word and include them into the suggested correction. To change *conduct survey* to *conduct **a** survey*, the annotator could change *conduct* to *conduct a*, change *survey* to *a survey*, or change the whole phrase *conduct survey* into *conduct **a** survey*. If we only compare the exact text spans selected by the annotators when measuring agreement, these different ways to select the context could easily cause us to conclude that the annotators disagree when they in fact agree on the corrected phrase. This would lead to an underestimation of annotator agreement. To address this problem, we perform a simple text span normalization. First, we "grow" the selected context to align with whitespace boundaries. For example, if an annotator just selected the last character *e* of the word *use* and provided *ed* as a correction, we grow this annotation so that the whole word *use* is selected and *used* is the correction. Second, we tokenize the text and "trim" the context by removing tokens at the start and end that are identical in the original and the correction. Finally, the annotations are "projected" onto the individual tokens they span, i.e., an annotation that spans a phrase of multiple tokens is broken up into multiple token-level annotations. Now, we can compare two annotations at the token level in a meaningful way. Here is a tokenized example sentence from the annotator agreement study with annotations from two annotators.

Source	:	This phenomenon opposes the real .
Annotator A	:	This phenomenon opposes (the $\rightarrow \epsilon$ (ArtOrDet))
		(real $\rightarrow$ reality (Wform)) .
Annotator B	:	This phenomenon opposes the (real $\rightarrow$ reality (Wform)) .

Annotator A and B agree that the first three words *This, phenomenon,* and *opposes* and the final period are correct and do not need any correction. The annotators also agree that the word *real* is part of a word form (Wform) error and should be replaced with *reality*. However, they disagree with respect to the article *the*: annotator A believes there is an article error (ArtOrDet) and that the article has to be deleted while annotator B believes that the article is acceptable in this position.

The example has shown that annotator agreement can be measured with respect to three different criteria: whether there is an error, what type of error it is, and how the error should be corrected. Accordingly, we analyze annotator agreement under three different conditions:

- **Identification** Agreement of tagged tokens regardless of error category.
- **Classification** Agreement of error category, given identification.
- **Exact** Agreement of error category and correction, given identification.

In the identification task, we are interested to see how well annotators agree on whether something is a grammatical error or not. In the example above, annotators A and B agree on 5 out of 6 tokens and disagree on one token (*the*). That results in an identification agreement of $5/6 = 83\%$. In the classification task, we investigate how well annotators agree on the type of error, given that both have tagged the token as an error. In the example, the classification agreement is 100% as both annotator A and B tagged the word *real* as a word form (Wform) error. Finally, for the exact task annotators are considered to agree if they agree on the error category *and* the correction given that they both have tagged the token as an error. In the example, the classification agreement is 100% as both annotators give the same error category Wform and the same correction *reality* for the word *real*. We use the popular Cohen's Kappa coefficient (Cohen, 1960) to measure annotator agreement between annotators. Cohen's Kappa is defined as

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)} \tag{3.1}$$

where $Pr(a)$ is the probability of agreement and $Pr(e)$ is the probability of chance agreement. We can estimate $Pr(a)$ and $Pr(e)$ from the double annotated essays through maximum-likelihood estimation. For two annotators A and B, the probability of agree-

Annotator 1	**Annotator 2**	**Kappa-iden**	**Kappa-clas**	**Kappa-exact**
A	B	0.4775	0.6206	0.5313
A	C	0.3627	0.5352	0.4956
B	C	0.3230	0.4894	0.4246
Average		0.3877	0.5484	0.4838

Table 3.2: Cohen's Kappa coefficients for annotator agreement.

ment is

$$Pr(a) = \frac{\text{\#agreed tokens}}{\text{\#total tokens}} \tag{3.2}$$

where #agreed tokens are counted as described in the last section and #total tokens is the total number of tokens in the subset of jointly annotated documents. The probability of chance agreement is computed as

$$\begin{aligned} Pr(e) &= Pr(A=1, B=1) + Pr(A=0, B=0) \\ &= Pr(A=1) \times Pr(B=1) + Pr(A=0) \times Pr(B=0) \end{aligned} \tag{3.3}$$

where $Pr(A = 1)$ and $Pr(A = 0)$ symbolize the events of annotator A tagging a token as "error" or "no error" respectively. We make use of the assumption that both annotators perform the task independently. $Pr(A = 1)$ and $Pr(A = 0)$ can be computed through maximum-likelihood estimation.

$$Pr(A=1) = \frac{\text{\# annotated tokens of annotator A}}{\text{\# total tokens}} \tag{3.4}$$

$$Pr(A=0) = \frac{\text{\# unannotated tokens of annotator A}}{\text{\# total tokens}} \tag{3.5}$$

The probabilities $Pr(B = 1)$ and $Pr(B = 0)$ are computed analogously. The chance agreement for this task is quite high, as the number of not annotated tokens is much higher than the number of annotated tokens. The Cohen's Kappa coefficients for the three annotators and the average Kappa coefficient are listed in Table 3.2. We observe that the Kappa scores are relatively low and that there is a substantial amount of variability in the Kappa coefficients; annotator A and B show a higher agreement with each other than they do with annotator C. According to Landis and Koch (1977), Kappa

“Public spending on the aged should be limited so that money can be diverted to other areas of the country’s development.” Do you agree?
Surveillance technology such as RFID (radio-frequency identification) should not be used to track people (e.g. human implants and RFID tags on people or products). Do you agree? Support your argument with concrete examples.
Choose a concept or prototype currently in research and development and not widely available in the market. Present an argument on how the design can be improved to enhance safety. Remember to consider influential factors such as cost or performance when you summarize and rebut opposing views. You will need to include very recently published sources in your references.

Table 3.3: Example question prompts from the NUCLE corpus.

scores between 0.21 and 0.40 are considered fair, and scores between 0.41 and 0.60 are considered moderate. The average Kappa score for identification can therefore only be considered fair and the Kappa scores for classification and exact agreement are moderate. Thus, a first interesting result of the pilot study was that annotators find it harder to agree on whether a word is grammatically correct than agreeing on the type of error or how it should be corrected. As a summary the annotator agreement study shows that grammatical error correction, especially grammatical error identification, is a difficult problem.

3.1.3 Data Collection and Annotation

The main data collection for the NUCLE corpus took place between August and December 2009. We collected a total of 2,249 student essays from 6 English courses at CELC. The courses are for students who need language support for their academic studies. The essays were written as course assignments on a wide range of topics, like technology innovation or health care. Some example question prompts are shown in Table 3.3 Student would typically have to write two essays assignments during one course. The length of each essay was supposed to be around 500 words, although most essays were longer than the required length. From this data set, a team of 10 CELC instructors annotated 1,414 essays with over 1.2 million words between October 2009 and April 2010. Due to budget constraints, we were unfortunately not able to perform double annotations for the main corpus. Annotators were asked to label an error with

NUS Corpus of Learner English	
Documents	1,414
Sentences	59,871
Average sentences per document	42.34
Word tokens	1,220,257
Average word tokens per document	862.98
Average word tokens per sentence	20.38
Word types	30,492
Error annotations	46,597
Average error annotations per document	32.95
Error annotations per 100 word tokens	3.82

Table 3.4: Overview of the NUCLE corpus

more than one error tag, if applicable. The results of the annotation exercise were a total of 46,597 error tags. The essays and the annotations were released as the NUCLE corpus through the NUS Enterprise R2M portal on 1 July 2011. The link to the corpus can be found on the NLP group's website[1].

3.1.4 NUCLE Corpus Statistics

This section provides basic statistics about the corpus and the collected annotations. These statistics already reveal some interesting insights about the nature of grammatical errors in learner text. In particular, we are interested in the questions of how frequent errors are in the corpus and the most frequent error categories. The basic statistics of the NUCLE corpus are shown in Table 3.4. We can see that grammatical errors are very *sparse*, even in learner text. In the NUCLE corpus there are 46,597 annotated errors in a corpus with 1,220,257 word tokens. That makes an error density of 3.82 errors per hundred words. In other words, over 96% of the word tokens in the corpus are grammatically correct. This shows that the students whose essays were utilized for the corpus already have a relative high proficiency of English. But it also means that performing automatic grammatical error correction requires the computer algorithm to find the proverbial needle in the hay stack of correct words, and once the needle is found the algorithm still needs to determine the correct type of the error and the correction. When we look at the distribution of errors across documents, we can make another in-

[1]http://nlp.comp.nus.edu.sg/corpora

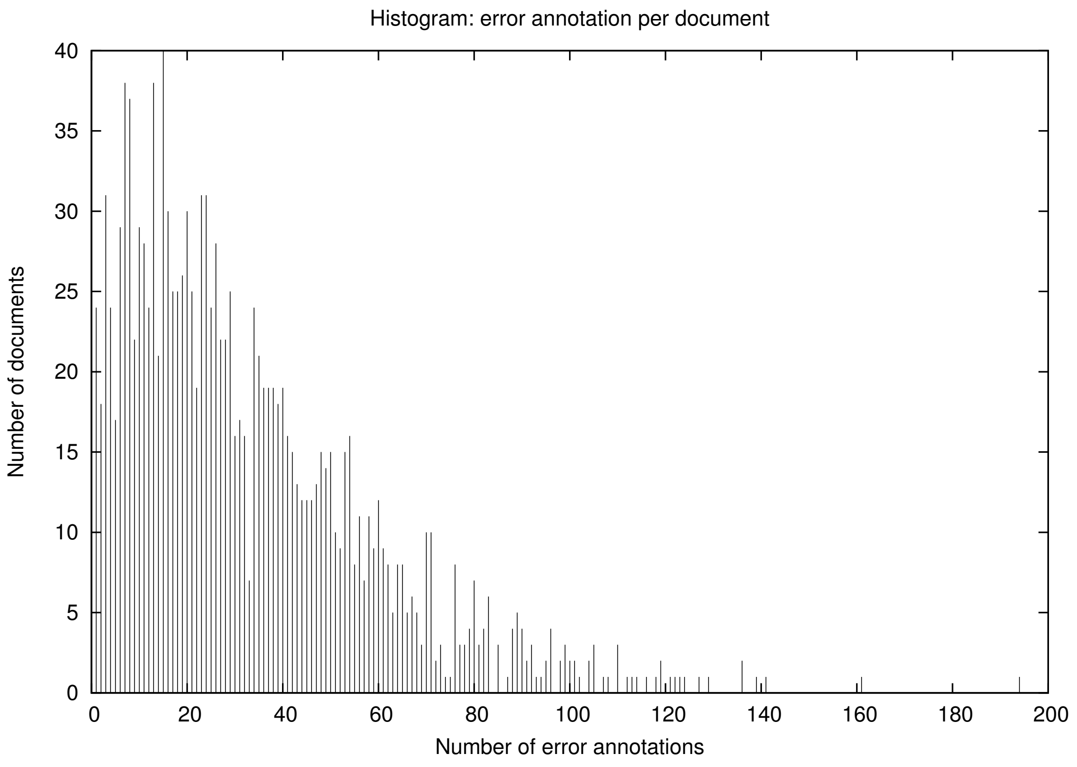


Figure 3.2: Histogram of error annotations per document in NUCLE.

teresting observation. Figure 3.2 shows a histogram of the number of error annotations per document. The distribution appears non-Gaussian and is heavily skewed to the left while some documents have significantly more errors than the average document. That means that although grammatical errors are rare *in general* there are also documents with many error annotations. 32 documents have more than 100 error annotations and the highest number of error annotations in a document is 194. The mode, i.e., the most frequent value in the histogram, is 15 which is to the left of the average of 32.95. A similar pattern can be observed when we look at the distribution of errors per sentence. Figure 3.3 shows a histogram of the number of error annotations per sentence in NUCLE. For this histogram, only the error annotations which start and end within sentence boundaries are considered. The histogram shows that 57.64% of all sentences have zero errors, 20.48% have exactly one error, and 10.66% have exactly two errors, and 11.21% of all sentences have more than two errors. Although the frequency decreases quickly for higher error counts, the highest observed number of error annotations for a sentence is 28.

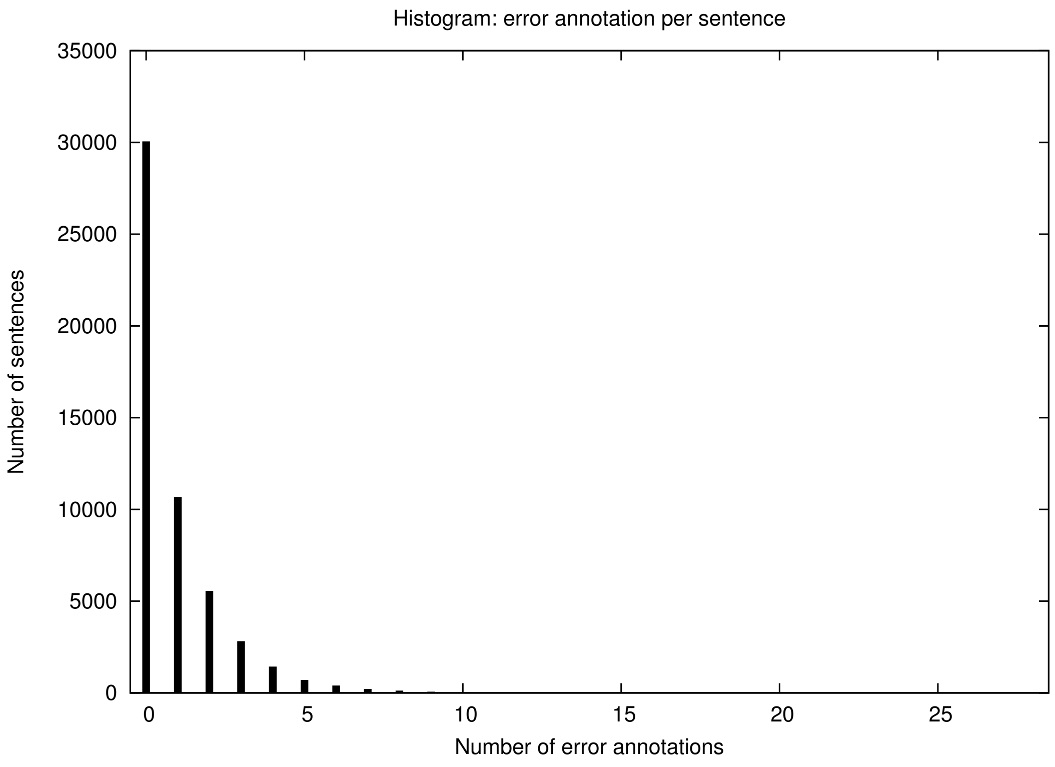


Figure 3.3: Histogram of error annotations per sentence in NUCLE.

The skewed distribution of errors in the NUCLE corpus is an interesting observation. A possible explanation for the long tail of the distribution could be a "rich-get-richer" type of dynamics: if a learner has made a lot of mistakes in her essay so far, the chances of her making more errors in the remainder of the essay increases, for example because she makes systematic errors which are likely to be repeated. However, the distributions do not seem to exhibit a classical power-law distribution either. Explaining the cognitive processes that produce the observed error distribution is beyond the scope of this thesis, but it would certainly be an interesting question to investigate.

So far, we have only been concerned with how many errors learners make overall. But it is also important to understand what types of errors language learners make. Error categories that appear more frequently should be addressed with higher priority when creating an automatic error correction system. Figure 3.4 shows a histogram of error categories. Again, we can observe a skewed distribution with a few error categories being very frequent and many error categories being comparatively infrequent. The top five error categories are wrong collocation/idiom/preposition (Wcip)

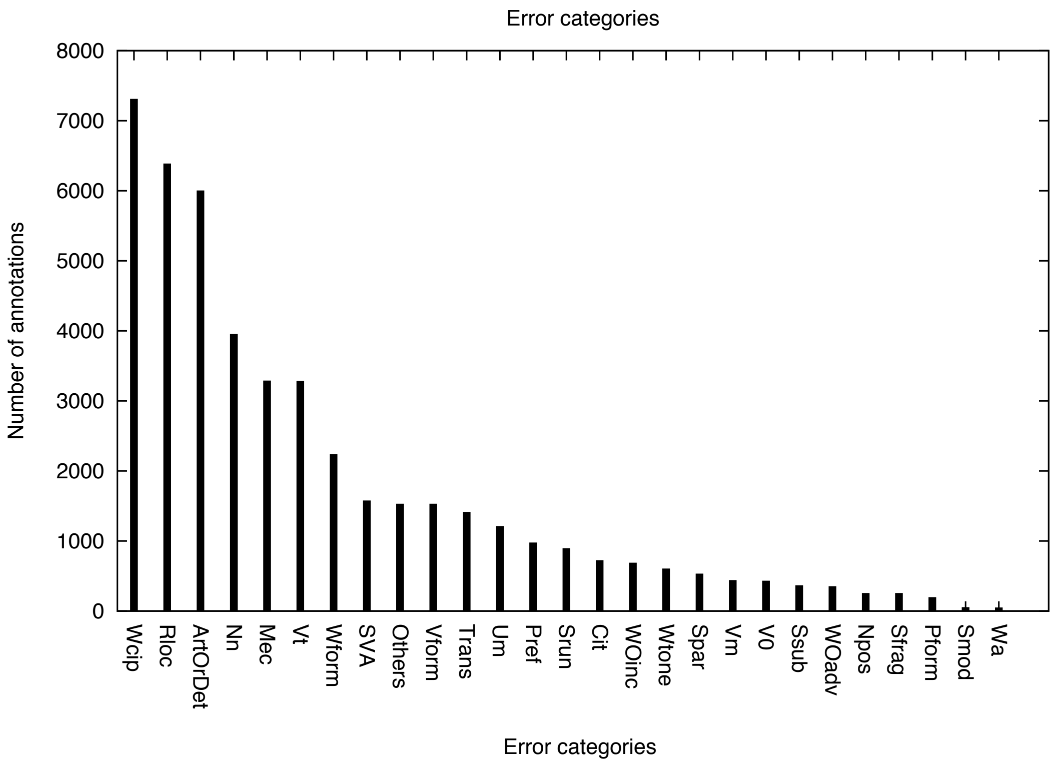


Figure 3.4: Error categories histogram for the NUCLE corpus.

with 7,312 instances or 15.69% of all annotations, local redundancies (Rloc) (6390 instances, 13.71%), article or determiner (ArtOrDet) (6004 instances, 12.88%), noun number (Nn) (3955 instances, 8.49%), and mechanics (Mec) (3290 instances, 7.06%). These top five error categories account for 57.83% of all error annotations. The next 5 categories are verb tense (Vt) (3288 instances, 7.06%) word form (Wform) (2241 instances, 4.81%), subject-verb agreement (SVA) (1578 instances, 3.38%), other errors that could not be grouped into any of the error categories (1532 instances, 3.29%), and Verb form (Vform) (1416, 3.04%). Together, the top 10 error categories account for 79.66% of all annotated errors. A manual inspection showed that a large percentage of the local redundancy errors involve articles that are deemed redundant by the annotator and should be deleted. These errors could also be considered article or determiner errors. For the Wcip errors, we observed that most Wcip errors are preposition errors. This confirms that articles and prepositions are the two most frequent error categories for EFL learners (Leacock et al., 2010). We will return to article and preposition errors in the next chapter.

In this section, we have presented the NUS Corpus of Learner English. Apart from being used in our own research on article and preposition errors (Section 4) and lexical choice errors (Section 5), the corpus is being used in the CoNLL-2013 Shared Task on grammatical error correction (Ng et al., 2013).

3.2 Helping Our Own data sets

The two Helping Our Own (HOO) shared tasks organized by Robert Dale and Adam Kilgariff (Dale and Kilgarriff, 2011; Dale et al., 2012) were the first shared evaluation campaigns for grammatical error correction. Both tasks released annotated data sets for system development and evaluation.

The HOO 2011 data set consists of papers written by non-native authors of English within the natural language processing community. The goal of HOO was to develop authoring tools that help non-native English speaking members of the NLP community to write better papers (hence the name of the shared task). The documents for HOO 2011 were extracted from 19 papers that had previously been published in a conference or workshop organized by the Association for Computational Linguistics from the ACL anthology. The criteria for selection were that the paper appeared to be written by a non-native speaker (based on the first author name) and seemed to have a relatively high amount of grammatical errors. Two parts of each paper (called *fragments*) were annotated by two editors (although no double annotation was performed). One fragment of each document was released with the training data, the other one was kept as part of the test data. Each extracted fragment contains about 1,000 words. The 19 released files in the training data contain about 22,00 words. The test data contains 18 files (one fragment had to be excluded by the organizers) and about 18,000 tokens in total. The HOO 2011 test data is available at shared task's Google group[2] after registration.

The HOO 2012 shared task shifted the goal of the task from error correction for NLP researchers to grammar correction for language learners in general. Accordingly, the domain of the data shifted as well. The training data in HOO 2012 is a subset of the

[2]`http://groups.google.com/group/hoo-nlp/`

Cambridge Learner Corpus FCE (First Certificate in English) data set that was made available one year earlier by Yannakoudakis *et al.* (2011) from Cambridge University Press. The FCE corpus contains exam essays written by students taking the First Certificate in English Examination. The FCE corpus consists of 1,244 documents with about 420,000 words. Of these, 1000 documents were selected as the HOO 2012 training data. The rest was withheld to serve as test data in case the organizers would not be able to obtain new, unseen test documents from Cambridge University press. Eventually, the organizers succeeded to obtain a test set of 100 new, unseen documents. Although no new training data was released, the complete FCE corpus could be used by participants for training purposes. The HOO 2012 training and test data only contains annotations for determiner and preposition errors which were the focus of the task.

3.3 Evaluation for Grammatical Error Correction

Progress in natural language processing research is driven and measured by automatic evaluation methods. Automatic evaluation allows fast and inexpensive feedback during development, and objective and reproducible evaluation during testing time. For grammatical error correction, an automatic evaluation metric needs to automatically compute some notion of similarity between the corrections proposed by a computer algorithm and a gold-standard reference provided by a human expert. The more closely the corrections of the algorithm match the human expert, the better the algorithm is. A good evaluation metric should also be simple to compute and interpretable for the human evaluator.

A naive way to evaluate grammatical error correction would be to compute accuracy which is defined as the ratio of the total number of correct predictions by an algorithm and the total number of predictions.

$$\text{accuracy} = \frac{\text{\# correct predictions}}{\text{\# total predictions}} \tag{3.6}$$

Accuracy is a good evaluation metric when evaluating a task where every instance is equally relevant to the evaluation score, for example, when testing how well an algorithm can re-predict the original prepositions in non-learner text after all prepositions were removed. However, for the evaluation of grammatical error correction algorithms on learner text, accuracy is not well suited. Consider a preposition correction task with 100 prepositional phrases in the test set. Grammatical errors are sparse, so let us assume that out of the 100 prepositional phrases ten contain a grammatical error. A simple "do nothing" baseline that leaves all the prepositions unchanged would achieve 90% accuracy. Another hypothetical algorithm changes 20 prepositions: five preposition errors are successfully corrected, another five preposition errors are detected but not successfully corrected, and another ten correct prepositions are accidentally changed to wrong prepositions. The remaining 80 prepositions are kept unchanged. The second algorithm would achieve $(80 + 5)/100 = 85\%$ accuracy, five percent less than the "do nothing" baseline even though the second algorithm actually corrects half of the errors in the test set. The high score for the "do nothing" baseline does not meet our intuition about grammatical error correction, an algorithm that does nothing should receive a score of zero, and an algorithm that corrects all the errors and leaves all other words unchanged should achieve a perfect score. The reason for the unintuitive evaluation result is that accuracy weighs all test instances equally. We would think that a correct preposition that was left unchanged should contribute less to the score than a erroneous preposition that was successfully corrected. Because a large portion of the test instances in grammatical error correction fall into the "no correction required" category, we need an evaluation metric that takes this into account.

3.3.1 Precision, Recall, F$_1$ Score

F_1 score is a popular metric in natural language processing and information retrieval that fulfills all of these criteria and is particularly suitable for situations where only a subset of the instances is of interest. To define F_1 score for error correction, we first need to define the precision and recall of an error correction algorithm. The *precision* of an algorithm is defined as the proportion of suggested corrections that agree with the human annotator with respect to the total number of proposed corrections. It is easily computed as the number of correct changes proposed divided by the total number of proposed changes.

$$\text{precision} = \frac{\text{\# correct changes}}{\text{\# total proposed changes}} \tag{3.7}$$

The *recall* of an algorithm is defined as the proportion of suggested corrections that agree with the human annotator with respect to the total number of corrections in the gold-standard reference. It is computed as the number of correct changes divided by the total number of changes in the gold-standard reference.

$$\text{recall} = \frac{\text{\# correct changes}}{\text{\# total changes in gold standard}} \tag{3.8}$$

There is typically a trade off between precision and recall. If an algorithm is very conservative and only proposes corrections in cases where it is extremely confident, it is more likely to get high precision but recall will be lower. If the algorithm is extremely aggressive in proposing corrections, it is more likely to get higher recall at the expense of lower precision. F_1 score combines precision and recall into a single number. It is defined as the harmonic mean between precision and recall.

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{3.9}$$

F_1 score weighs precision and recall equally but in general, F score can be computed with different weights for precision and recall (van Rijsbergen, 1979). Note that precision, recall, and F_1 score only take into account those test instances where either the algorithm predicted a correction or there is a correction annotated in the gold standard. All instance that are not tagged as errors by either the algorithm or the gold standard do not affect the evaluation. An algorithm that leaves all words unchanged will receive a recall of zero and consequently an F_1 score of zero, in line with our intuition. F_1 score is the most common metric for grammatical error correction as it is easy to understand, intuitive, and interpretable (Leacock et al., 2010).

3.4 MaxMatch Method for Evaluation

To compute precision, recall, and F_1 score, it is necessary for the evaluation method to first determine what corrections the correction algorithm has actually performed. A correction, or *edit*, minimally consists of a start and end position in the original text and a word or phrase that should replace the original word or phrase used by the writer. Unfortunately, this process is more complicated than one might expect because the set of edit operations that transforms the original source sentence into the corrected system output sentence is ambiguous. This is due to two reasons. First, the set of edits that transforms one string into another is not necessarily unique, even at the token level. Second, edits can consist of longer phrases which introduce additional ambiguity. To see how this can affect evaluation, consider the following source sentence and system hypothesis from the Helping Our Own 2011 (HOO 2011) shared task (Dale and Kilgarriff, 2011) on grammatical error correction:

Source :	Our baseline system feeds word into PB-SMT pipeline.
Hypot. :	Our baseline system feeds **a** word into PB-SMT pipeline.

The HOO evaluation script extracts the system edit ($\epsilon \rightarrow$ a), i.e., inserting the article *a*. Unfortunately, the gold-standard annotation instead contains the edits (word $\rightarrow$ $\{$a word, words$\}$). Although the extracted system edit results in the *same* corrected sentence as the first gold-standard edit option, the system hypothesis was considered to be invalid. Even if the participants were asked to submit a set of system edits instead of the corrected system output, the problem would still persist. Without knowing how the text is annotated in the gold standard, it is impossible to know which of the possible edits (($\epsilon \rightarrow$ a), (a $\rightarrow$ a word), (feeds $\rightarrow$ feeds a), . . .) to submit.

In this section, we propose a method, called *MaxMatch* (M^2), to overcome this problem. The key idea is that if there are multiple possible ways to arrive at the same correction, the system should be evaluated according to the set of edits that matches the gold-standard as often as possible. To this end, we propose an algorithm for efficiently computing the set of phrase-level edits with the maximum overlap with the gold standard. The edits are subsequently scored using F_1 score. We test our method in the context of the HOO 2011 shared task and show that our method results in a more accurate evaluation for error correction. This method was first presented in (Dahlmeier and Ng, 2012b). The M^2 scorer is available for download at `http://nlp.comp.nus.edu.sg/software/`.

3.4.1 Method

We begin by establishing some notation. Let us consider a set of *source sentences* $S = \{\mathbf{s}_1, \ldots, \mathbf{s}_n\}$ together with a set of *hypotheses* $H = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$ generated by an error correction system. Let $G = \{\mathbf{g}_1, \ldots, \mathbf{g}_n\}$ be the set of gold standard annotations for the same sentences. Each annotation $\mathbf{g}_i = \{g_i^1, \ldots, g_i^r\}$ is a set of *edits*. An edit is a triple (a, b, C), consisting of:

- start and end (token-) offsets a and b with respect to a source sentence,
- a correction C. For gold-standard edits, C is a set containing one or more possible corrections. For system edits, C is a single correction.

Evaluation of the system output involves the following two steps:

1. Extracting a set of *system edits* $\mathbf{e}_i$ for each source-hypothesis pair $(\mathbf{s}_i, \mathbf{h}_i)$.
2. Evaluating the system edits for the complete test set with respect to the gold standard G.

The remainder of this section describes a method for solving these two steps. We start by describing how to construct an *edit lattice* from a source-hypothesis pair. Then, we show that finding the optimal sequence of edits is equivalent to solving a shortest path search through the lattice. Finally, we describe how to evaluate the edits using F_1 score.

Edit Lattice

We start from the well-established Levenshtein distance (Levenshtein, 1966), which is defined as the minimum number of insertions, deletions, and substitutions needed to

		Our	baseline	system	feeds	a	word	into	PB-SMT	pipeline	.
	0	1	2	3	4	5	6	7	8	9	10
Our	1	0	1	2	3	4	5	6	7	8	9
baseline	2	1	0	1	2	3	4	5	6	7	8
system	3	2	1	0	1	2	3	4	5	6	7
feeds	4	3	2	1	0	1	2	3	4	5	6
word	5	4	3	2	1	1	1	2	3	4	5
into	6	5	4	3	2	2	2	1	2	3	4
PB-SMT	7	6	5	4	3	3	3	2	1	2	3
pipeline	8	7	6	5	4	4	4	3	2	1	2
.	9	8	7	6	5	5	5	4	3	2	1

Figure 3.5: The Levenshtein matrix and the shortest path for a source sentence "Our baseline system feeds word into PB-SMT pipeline ." and a hypothesis "Our baseline system feeds a word into PB-SMT pipeline ."

transform one string into another. The Levenshtein distance between a source sentence $\mathbf{s}_i = s_i^1, \dots, s_i^k$ and a hypothesis $\mathbf{h}_i = h_i^1, \dots, h_i^l$ can be efficiently computed using a two dimensional matrix that is filled using a classic dynamic programming algorithm. We assume that both $\mathbf{s}_i$ and $\mathbf{h}_i$ have been tokenized. The matrix for the example in this section is shown in Figure 3.5. By performing a simple breadth-first search, similar to the Viterbi algorithm, we can extract the lattice of all shortest paths that lead from the top-left corner to the bottom-right corner of the Levenshtein matrix. Each vertex in the lattice corresponds to a cell in the Levenshtein matrix, and each edge in the lattice corresponds to an atomic edit operation: inserting a token, deleting a token, substituting a token, or leaving a token unchanged. Each path through the lattice corresponds to a shortest sequence of edits that transform $\mathbf{s}_i$ into $\mathbf{h}_i$. We assign a unit cost to each edge in the lattice.

We have seen that annotators can use longer phrases and that phrases can include unchanged words from the context, e.g., the gold edit from the example is

$$(4, 5, \text{word}, \{\text{a word}, \text{words}\}).$$

However, it seems unrealistic to allow an arbitrary number of unchanged words in an edit. In particular, we want to avoid very large edits that cover complete sentences. Therefore, we limit the number of unchanged words by a parameter u. To allow for

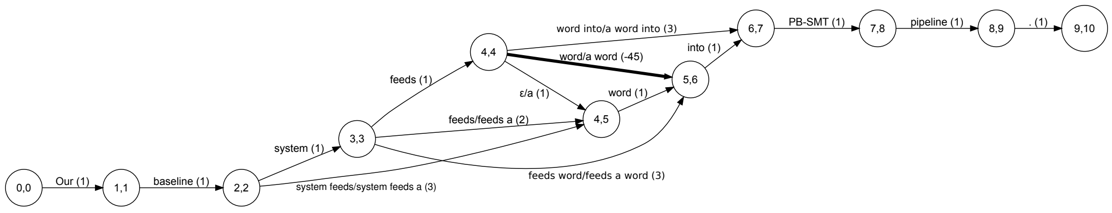


Figure 3.6: The edit lattice for "Our baseline system feeds ($\epsilon \rightarrow$ a) word into PB-SMT pipeline ." Edge costs are shown in parentheses. The edge from (4,4) to (5,6) matches the gold annotation and carries a negative cost.

phrase-level edits, we add transitive edges to the lattice as long as the number of unchanged words in the newly added edit is not greater than u and the edit changes at least one word. Let $e_1 = (a_1, b_1, C_1)$ and $e_2 = (a_2, b_2, C_2)$ be two edits corresponding to adjacent edges in the lattice, with the first end offset b_1 being equal to the second start offset a_2. We can combine them into a new edit $e_3 = (a_1, b_2, C_1 + C_2)$, where $C_1 + C_2$ is the concatenation of strings C_1 and C_2. The cost of a transitive edge is the sum of the costs of its parts. The lattice extracted from the example sentence is shown in Figure 3.6.

Finding Maximally Matching Edit Sequence

Our goal is to find the sequence of edits $\mathbf{e}_i$ with the maximum overlap with the gold standard. Let $L = (V, E)$ be the edit lattice graph from the last section. We change the cost of each edge whose corresponding edit has a match in the gold standard to $-(u + 1) \times |E|$. An edit e *matches* a gold edit g iff they have the same offsets and e's correction is included in g:

$$match(e, g) \Leftrightarrow e.a = g.a \wedge e.b = g.b \wedge e.C \in g.C \qquad (3.10)$$

Then, we perform a single-source shortest path search with negative edge weights from the start to the end vertex.[3] This can be done efficiently, for example with the Bellman-Ford algorithm (Cormen et al., 2001). As the lattice is acyclic, the algorithm is guaran-

[3]To break ties between non-matching edges, we add a small cost $\zeta \ll 1$ to all non-matching edges, thus favoring paths that use fewer edges, everything else being equal.

teed to terminate and return a shortest path.

Theorem 1. *The set of edits corresponding to the shortest path has the maximum overlap with the gold standard annotation.*

Proof. Let $\mathbf{e} = e^1, \ldots, e^k$ be the edit sequence corresponding to the shortest path and let p be the number of matched edits. Assume that there exists another edit sequence $\mathbf{e}'$ with higher total edge weights but $p' > p$ matching edits. Then we have

$$\begin{aligned} p(-(u+1)|E|) + q &\leq p'(-(u+1)|E|) + q' \\ \Leftrightarrow (q - q') &\leq (p' - p)(-(u+1)|E|), \end{aligned} \tag{3.11}$$

where q and q' denote the combined cost of all non-matching edits in the two paths, respectively. Because $p' - p \geq 1$, the right hand side is at most $-(u+1)|E|$. Because q and q' are positive and bounded by $(u+1)|E|$, the left hand side cannot be smaller than or equal to $-(u+1)|E|$. This is a contradiction. Therefore there cannot exist such an edit sequence $\mathbf{e}'$, and $\mathbf{e}$ is the sequence with the maximum overlap with the gold-standard annotation. □

Evaluating Edits

What is left to do is to evaluate the set of edits with respect to the gold standard. This is done by computing precision, recall, and F_1 score as defined in Equations 3.7, 3.8, and 3.9 in the previous section. Let the set of system edits be $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$ and let the set of gold edits be $\{\mathbf{g}_1, \ldots, \mathbf{g}_n\}$. Precision, recall, and F_1 score for all sentences are computed as

$$P = \frac{\sum_{i=1}^{n} |\mathbf{e}_i \cap \mathbf{g}_i|}{\sum_{i=1}^{n} |\mathbf{e}_i|} \tag{3.12}$$

$$R = \frac{\sum_{i=1}^{n} |\mathbf{e}_i \cap \mathbf{g}_i|}{\sum_{i=1}^{n} |\mathbf{g}_i|} \tag{3.13}$$

$$F_1 = 2 \times \frac{P \times R}{P + R}, \tag{3.14}$$

Team	HOO scorer			M^2 scorer		
	P	R	F_1	P	R	F_1
JU (0)	10.39	3.78	5.54	12.30	4.45	6.53
LI (8)	20.86	3.22	5.57	21.12	3.22	5.58
NU (0)	29.10	7.38	11.77	31.09	7.85	12.54
UI (1)	50.72	13.34	21.12	54.61	14.57	23.00
UT (1)	5.01	4.07	4.49	5.72	4.45	5.01

Table 3.5: Results for participants in the HOO 2011 shared task. The run of the system is shown in parentheses.

where we define the intersection between $\mathbf{e}_i$ and $\mathbf{g}_i$ as

$$\mathbf{e}_i \cap \mathbf{g}_i = \{e \in \mathbf{e}_i \mid \exists\, g \in \mathbf{g}_i(match(e, g))\}. \qquad (3.15)$$

3.4.2 Experiments and Results

We experimentally test our M^2 method in the context of the HOO 2011 shared task. We test our method by re-scoring the best runs of the participating teams[4] in the HOO 2011 shared task with the M^2 scorer and comparing the scores with the official HOO 2011 scorer, which simply uses GNU `wdiff`[5] to extract system edits. We obtain each system's output and segment it at the sentence level according to the gold standard sentence segmentation. The source sentences, system hypotheses, and corrections are tokenized using the Penn Treebank standard (Marcus et al., 1993). The character edit offsets are automatically converted to token offsets. We set the parameter u to 2, allowing up to two unchanged words per edit. The results are shown in Table 3.5. Note that the M^2 scorer and the HOO 2011 scorer adhere to the same score definition and only differ in the way the system edits are computed. We can see that the M^2 scorer results in higher scores than the official scorer for all systems, showing that the official scorer missed some valid edits. For example, the M^2 scorer finds 155 valid edits for the UI system compared to 141 found by the official scorer, and 83 valid edits for the NU system (our submission), compared to 78 by the official scorer. We manually inspect the output of the scorers and find that the M^2 scorer indeed extracts the correct edits matching the

[4]Except one team that did not submit any plain text output.

[5]`http://www.gnu.org/s/wdiff/`

M^2 scorer	… should basic translational unit be (word $\rightarrow$ a word) …
HOO scorer	… should basic translational unit be *($\epsilon \rightarrow$ a) word …
M^2 scorer	… development set similar (with $\rightarrow$ to) ($\epsilon \rightarrow$ the) test set …
HOO scorer	… development set similar *(with $\rightarrow$ to the) test set …
M^2 scorer	($\epsilon \rightarrow$ The) *(Xinhua portion of $\rightarrow$ xinhua portion of) the English Gigaword3 …
HOO scorer	*(Xinhua $\rightarrow$ The xinhua) portion of the English Gigaword3 …

Table 3.6: Examples of different edits extracted by the M^2 scorer and the official HOO scorer. Edits that do not match the gold-standard annotation are marked with an asterisk (*).

gold standard where possible. Examples are shown in Table 3.6.

3.4.3 Discussion

The evaluation framework proposed in this work differs slightly from the one in the HOO 2011 shared task.

Sentence-by-sentence. We compute the edits between source-hypothesis sentence pairs, while the HOO 2011 scorer computes edits at the document level. As the HOO 2011 data comes in a sentence-segmented format, both approaches are equivalent, while sentence-by-sentence is easier to work with.

Token-level offsets. In this thesis, we consider the start and end of an edit as *token offsets*, while the HOO 2011 data uses character offsets. Character offsets make the evaluation procedure very brittle as a small change, e.g., an additional whitespace character, will affect all subsequent edits. Character offsets also introduce ambiguities in the annotation, e.g., whether a comma is part of the preceding token.

Alternative scoring. The HOO 2011 shared task defines three different scores: *detection*, *recognition*, and *correction*. Effectively, all three scores are F_1 scores and only differ in the conditions on when an edit is counted as valid. Additionally, each score is reported under a "with bonus" alternative, where a system receives rewards for missed optional edits. The F_1 score defined in this chapter is equivalent to *correction without bonus*. The MaxMatch method can be used to compute detection and recognition scores and scores with bonus as well.

The M^2 scorer has been adopted as the official scorer for the CoNLL-2013 shared task on grammatical error correction (Ng et al., 2013).

3.5 Conclusion

In this chapter, we have presented the NUCLE learner corpus for grammatical error correction that we developed as part of this thesis. The corpus is currently the largest annotated learner data set that is available for research. We hope that this corpus will be a valuable resource for researchers for the quantitative analysis of learner errors and for training and testing better models on real, error-annotated text. We also described the data sets from the two HOO shared tasks. Finally, we presented a novel method, called MaxMatch (M^2), for evaluating grammatical error correction. Our method computes the sequence of phrase-level edits that achieves the highest overlap with the gold-standard annotation. Experiments on the HOO data show that our method overcomes deficiencies in the current evaluation method.

Chapter 4

Alternating Structure Optimization for Grammatical Error Correction

Despite the growing interest in grammatical error correction, research has been hindered until recently by the lack of a large annotated corpus of learner text that is available for research purposes. As a result, the standard approach to grammatical error correction has been to train an off-the-shelf classifier to re-predict words in non-learner text. Learning grammatical error correction models directly from annotated learner corpora is not well explored, as are methods that combine learner and non-learner text. Furthermore, the evaluation of grammatical error correction has been problematic. Previous work has either evaluated on artificial test instances as a substitute for real learner errors or on proprietary data that is not available to other researchers. As a consequence, existing methods have not been compared on the same test set, leaving it unclear where the current state of the art really is. With the availability of the NUCLE corpus that we have described in the last chapter, it is now possible to address these two problems: developing learning algorithms that take advantage of learner and non-learner data, and evaluating different methods on a large and publicly available set of real learner data.

In this chapter, we present a novel approach to grammatical error correction based on Alternating Structure Optimization (ASO) (Ando and Zhang, 2005). The approach is able to train models on annotated learner corpora while still taking advantage of large non-learner corpora. We conduct an extensive evaluation for article and preposition errors using six different feature sets proposed in previous work. We compare our proposed ASO method with two baselines trained on non-learner text and learner text, respectively. When these results were first published in (Dahlmeier and Ng, 2011b), they were the first extensive comparison of different feature sets on real learner text. Our experiments show that the proposed ASO algorithm significantly improves over both baselines. It also outperforms two commercial grammar checking software packages in a manual evaluation.

The remainder of this chapter is organized as follows. Section 4.1 describes the tasks. Section 4.2 formulates grammatical error correction as a classification problem. Section 4.3 extends this to the ASO algorithm. The experiments are presented in Section 4.4 and the results in Section 4.5. Section 4.6 contains a more detailed analysis of the results. Section 4.7 concludes this chapter.

4.1 Task Description

In this chapter, we focus on article and preposition errors, as they are among the most frequent types of errors made by EFL learners.

4.1.1 Selection vs. Correction Task

There is an important difference between training on annotated learner text and training on non-learner text, namely whether the observed word can be used as a feature or not. When training on non-learner text, the observed word cannot be used as a feature. The word choice of the writer is "blanked out" from the text and serves as the correct class. A classifier is trained to re-predict the word given the surrounding context. The *confusion set* of possible classes is usually pre-defined. This *selection task* formulation is convenient as training examples can be created "for free" from any text that is assumed to be free of grammatical errors. We define the more realistic *correction task* as follows: given a particular word and its context, propose an appropriate correction.

The proposed correction can be identical to the observed word, i.e., no correction is necessary. The main difference is that the word choice of the writer can be encoded as part of the features. The difference between the selection and correction task was first described by Rozovskaya and Roth (2010b).

4.1.2 Article Errors

For article errors, the classes are the three articles *a*, *the*, and the *null article* ϵ. This covers article insertion, deletion, and replacement errors. During training, each noun phrase (NP) in the training data is one training example. When training on learner text, the correct class is the article provided by the human annotator. When training on non-learner text, the correct class is the observed article. The context is encoded via a set of feature functions. During testing, each NP in the test set is one test example. The correct class is the article provided by the human annotator when testing on learner text or the observed article when testing on non-learner text.

4.1.3 Preposition Errors

The approach to preposition errors is similar to articles but typically focuses on preposition replacement errors. In this chapter, the classes are 36 frequent English prepositions (*about, along, among, around, as, at, beside, besides, between, by, down, during, except, for, from, in, inside, into, of, off, on, onto, outside, over, through, to, toward, towards, under, underneath, until, up, upon, with, within, without*), which we adopt from previous work (Tetreault, personal communication). Every prepositional phrase (PP) that is governed by one of the 36 prepositions is one training or test example. We ignore PPs governed by other prepositions.

4.2 Linear Classifiers for Error Correction

In this section, we formulate grammatical error correction as a classification problem and describe the feature sets for each task.

4.2.1 Linear Classifiers

We use classifiers to approximate the unknown relation between articles or prepositions and their contexts in learner text, and their valid corrections. The articles or prepositions and their contexts are represented as feature vectors $\mathbf{X} \in \mathcal{X}$. The corrections are the classes $Y \in \mathcal{Y}$.

In this work, we employ binary linear classifiers of the form $\mathbf{u}^T\mathbf{X}$ where $\mathbf{u}$ is a weight vector. The outcome is considered $+1$ if the score is positive and -1 otherwise. A popular method for finding $\mathbf{u}$ is *empirical risk minimization* with *least square regularization*. Given a training set $\{\mathbf{X}_i, Y_i\}_{i=1,\ldots,n}$, we aim to find the weight vector that minimizes the empirical loss on the training data

$$\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} \left(\frac{1}{n} \sum_{i=1}^{n} L(\mathbf{u}^T\mathbf{X}_i, Y_i) + \lambda \left|\left|\mathbf{u}\right|\right|^2 \right), \tag{4.1}$$

where $L(\hat{y}, y)$ is a loss function. We use a modification of Huber's robust loss function similar to that used in (Ando and Zhang, 2005)

$$L(\hat{y}, y) = \begin{cases} -4\hat{y}y & : \text{ if } \hat{y}y < -1 \\ (1 - \hat{y}y)^2 & : \text{ if } -1 \leq \hat{y}y < 1 \\ 0 & : \text{ if } \hat{y}y > 1. \end{cases} \tag{4.2}$$

We fix the regularization parameter λ to 10^{-4}. A multi-class classification problem with m classes can be cast as m binary classification problems in a *one-vs-rest* arrangement. We use the limited-memory, variable-metric method as implemented in the TAO optimization software (Munson et al., 2012) for the optimization.

The prediction of the classifier is the class with the highest score

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}} (\mathbf{u}_Y^T\mathbf{X}). \tag{4.3}$$

In earlier experiments, this linear classifier gave comparable or superior performance compared to a logistic regression classifier.

4.2.2 Features

We re-implement six feature extraction methods from previous work, three for articles and three for prepositions. The methods require different linguistic pre-processing: chunking, CCG parsing, and constituency parsing.

Article Errors

- **DeFelice** The system in (De Felice, 2008) for article errors uses a CCG parser to extract a rich set of syntactic and semantic features, including part of speech (POS) tags, hypernyms from WordNet (Fellbaum, 1998), and named entities.
- **Han** The system in (Han et al., 2006) relies on shallow syntactic and lexical features derived from a chunker, including the words before, in, and after the NP, the head word, and POS tags.
- **Lee** The system in (Lee, 2004) uses a constituency parser. The features include POS tags, surrounding words, the head word, and hypernyms from WordNet.

Preposition Errors

- **DeFelice** The system in (De Felice, 2008) for preposition errors uses a similar rich set of syntactic and semantic features as the system for article errors. In our re-implementation, we do not use a subcategorization dictionary, as this resource was not available to us.
- **TetreaultChunk** The system in (Tetreault and Chodorow, 2008b) uses a chunker to extract features from a two-word window around the preposition, including lexical and POS N-grams, and the head words from neighboring constituents.
- **TetreaultParse** The system in (Tetreault et al., 2010) extends (Tetreault and Chodorow, 2008b) by adding additional features derived from a constituency and a dependency parse tree.

For each of the above feature sets, we add the observed article or preposition as an additional feature when training on learner text.

4.3 Alternating Structure Optimization

This section describes the ASO algorithm and shows how it can be used for grammatical error correction.

4.3.1 The ASO Algorithm

Alternating Structure Optimization (Ando and Zhang, 2005) is a multi-task learning algorithm that takes advantage of the *common structure* of multiple related problems. Let us assume that we have m binary classification problems. Each classifier $\mathbf{u}_i$ is a weight vector of dimension p. Let Θ be an orthonormal $h \times p$ matrix that captures the common structure of the m weight vectors. We assume that each weight vector can be decomposed into two parts: one part that models the particular i-th classification problem and one part that models the common structure

$$\mathbf{u}_i = \mathbf{w}_i + \Theta^T \mathbf{v}_i. \tag{4.4}$$

The parameters $[\{\mathbf{w}_i, \mathbf{v}_i\}, \Theta]$ can be learned by *joint empirical risk minimization*, i.e., by minimizing the joint empirical loss of the m problems on the training data

$$\sum_{l=1}^{m} \left(\frac{1}{n} \sum_{i=1}^{n} L\left(\left(\mathbf{w}_l + \Theta^T \mathbf{v}_l\right)^T \mathbf{X}_i^l, Y_i^l \right) + \lambda \left|\left|\mathbf{w}_l\right|\right|^2 \right). \tag{4.5}$$

The key observation in ASO is that the problems used to find Θ do not have to be same as the *target problems* that we ultimately want to solve. Instead, we can automatically create *auxiliary problems* for the sole purpose of learning a better Θ.

Let us assume that we have k target problems and m auxiliary problems. We can obtain an approximate solution to Equation 4.5 by performing the following algorithm (Ando and Zhang, 2005):

1. Learn m linear classifiers $\mathbf{u}_i$ independently.
2. Let $U = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$ be the $p \times m$ matrix formed from the m weight vectors.
3. Perform Singular Value Decomposition (SVD) on U: $U = V_1 D V_2^T$. The first h column vectors of V_1 are stored as rows of Θ.
4. Learn $\mathbf{w}_j$ and $\mathbf{v}_j$ for each of the target problems by minimizing the empirical risk:
$$\frac{1}{n}\sum_{i=1}^{n} L\left(\left(\mathbf{w}_j + \Theta^T \mathbf{v}_j\right)^T \mathbf{X}_i, Y_i\right) + \lambda \left|\left|\mathbf{w}_j\right|\right|^2.$$
5. The weight vector for the j-th target problem is:
$$\mathbf{u}_j = \mathbf{w}_j + \Theta^T \mathbf{v}_j.$$

The number of retained eigenvectors h is a parameter which could be chosen through standard model selection techniques, like cross-validation. Ando and Zhang (2005) show that the ASO algorithm is in practice insensitive to the exact value of h. They use a fixed value $h = 50$ for all their experiments.

4.3.2 ASO for Grammatical Error Correction

The key observation in this chapter is that the selection task on non-learner text is a highly informative *auxiliary problem* for the correction task on learner text. For example, a classifier that can predict the presence or absence of the preposition *on* can be helpful for correcting wrong uses of *on* in learner text, e.g., if the classifier's confidence for *on* is low but the writer used the preposition *on*, the writer might have made a mistake. As the auxiliary problems can be created automatically, we can leverage the power of very large corpora of non-learner text.

Let us assume a grammatical error correction task with m classes. For each class, we define a binary auxiliary problem. The feature space of the auxiliary problems is a restriction of the original feature space $\mathcal{X}$ to all features except the observed word:

$\mathcal{X} \setminus \{X_{obs}\}$. The weight vectors of the auxiliary problems form the matrix U in Step 2 of the ASO algorithm from which we obtain Θ through SVD. As the number of auxiliary problems is not very large in our case, we retain all eigenvectors, thus eliminating the choice of the h parameter in practice. Given Θ, we learn the vectors $\mathbf{w}_j$ and $\mathbf{v}_j$, $j = 1, \ldots, k$ from the annotated learner text using the complete feature space $\mathcal{X}$.

This can be seen as an instance of *transfer learning* (Pan and Yang, 2010), as the auxiliary problems are trained on data from a different domain (non-learner text) and have a slightly different feature space ($\mathcal{X} \setminus \{X_{obs}\}$). We note that this method is general and can be applied to any classification problem in grammatical error correction.

4.4 Experiments

4.4.1 Data Sets

The main corpus in our experiments is the *NUS Corpus of Learner English (NUCLE)* that was presented in the last chapter. We use about 80% of the essays for training, 10% for development, and 10% for testing. We ensure that no sentences from the same essay appear in both the training and the test or development data. The motivation behind choosing as much as 80% of the essays for training is that we wish to have as much data as possible available for training. For the development and test data, we just require them to be large enough to allow for stable parameter tuning and statistically significant conclusions, respectively. As the NUCLE corpus contains 1.2 million words, the development and test set still contains several thousands of test instances which is sufficient for stable parameter tuning and statistically significant test results.

On average, only 1.8% of the articles and 1.3% of the prepositions in NUCLE contain an error. This figure is considerably lower compared to other learner corpora (Leacock et al., 2010, Ch. 3) and shows that the writers in the NUCLE corpus have a relatively high proficiency of English. We argue here that this makes the task considerably more difficult. Furthermore, to keep the task as realistic as possible, we do not filter the test data in any way.

In addition to NUCLE, we use a subset of the New York Times section of the Gigaword corpus[6] and the Wall Street Journal section of the Penn Treebank (Marcus et al., 1993) for some experiments.

4.4.2 Resources

We pre-process all corpora using the following tools: we use NLTK[7] for sentence splitting, OpenNLP[8] for POS tagging, YamCha (Kudo and Matsumoto, 2003) for chunking, the C&C tools (Clark and Curran, 2007) for CCG parsing and named entity recognition, and the Stanford parser (Klein and Manning, 2003a; Klein and Manning, 2003b) for constituency and dependency parsing.

4.4.3 Evaluation Metrics

For experiments on non-learner text, we report accuracy, which is defined as the number of correct predictions divided by the total number of test instances. For experiments on learner text, we report F_1 score between the correction proposed by the classifier and the gold-standard annotations as defined in Chapter 3.

4.4.4 Selection Task Experiments on WSJ Test Data

The first set of experiments investigates predicting articles and prepositions in non-learner text. This primarily serves as a reference point for the correction task described in the next section. We train classifiers as described in Section 4.2 on the Gigaword corpus. We train with up to 10 million training instances, which corresponds to about 37 million words of text for articles and 112 million words of text for prepositions. The test instances are extracted from section 23 of the WSJ and no text from the WSJ is included in the training data. The observed article or preposition choice of the writer is the class we want to predict. Therefore, the article or preposition cannot be part of

[6]LDC2009T13
[7]www.nltk.org
[8]opennlp.sourceforge.net

the input features. The proposed ASO method is not included in these experiments, as it uses the observed article or preposition as a feature which is only applicable when testing on learner text.

4.4.5 Correction Task Experiments on NUCLE Test Data

The second set of experiments investigates the primary goal of this chapter: to automatically correct grammatical errors in learner text. The test instances are extracted from NUCLE. In contrast to the previous selection task, the observed word choice of the writer can be different from the correct class and the observed word is available during testing. We investigate two different baselines and our ASO method.

The first baseline is a classifier trained on the Gigaword corpus in the same way as described in the selection task experiment. We use a simple thresholding strategy to make use of the observed word during testing. The system only flags an error if the difference between the classifier's confidence for its first choice and the confidence for the observed word is higher than a threshold t. The threshold parameter t is tuned on the NUCLE development data for each feature set. In our experiments, the value for t is between 0.7 and 1.2.

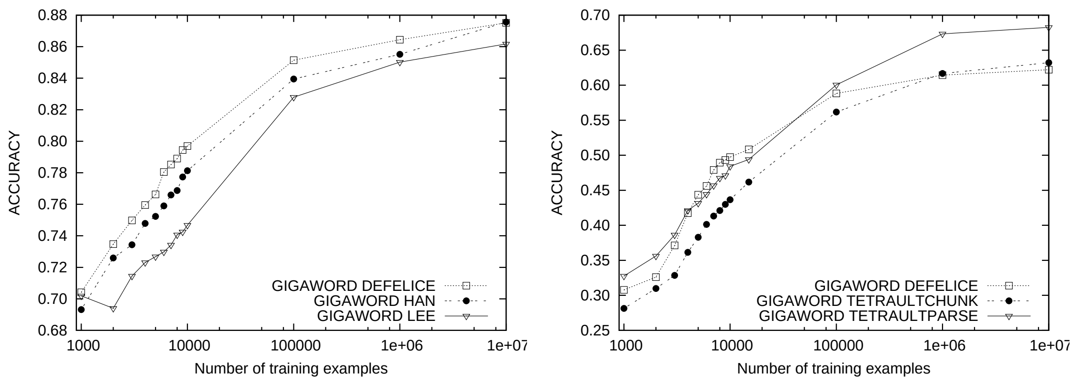
The second baseline is a classifier trained on NUCLE. The classifier is trained in the same way as the Gigaword model, except that the observed word choice of the writer is included as a feature. The correct class during training is the correction provided by the human annotator. As the observed word is part of the features, this model does not need an extra thresholding step. Indeed, we found that thresholding is harmful in this case. During training, the instances that do not contain an error greatly outnumber the instances that do contain an error. To reduce this imbalance, we keep all instances that contain an error and retain a random sample of q percent of the instances that do not contain an error. The undersample parameter q is tuned on the NUCLE development data for each data set. In our experiments, the value for q is between 20% and 40%.

The ASO method is trained in the following way. We create binary auxiliary problems for articles or prepositions, i.e., there are 3 auxiliary problems for articles and 36 auxiliary problems for prepositions. We train the classifiers for the auxiliary problems on the complete 10 million instances from Gigaword in the same ways as in the selection task experiment. The weight vectors of the auxiliary problems form the matrix U. We perform SVD to get $U = V_1 D V_2^T$. We keep all columns of V_1 to form Θ. The target problems are again binary classification problems for each article or preposition, but this time trained on NUCLE. The observed word choice of the writer is included as a feature for the target problems. We again undersample the instances that do not contain an error and tune the parameter q on the NUCLE development data. The value for q is between 20% and 40%. No thresholding is applied.

We also experimented with a classifier that is trained on the concatenated data from NUCLE and Gigaword. This model always performed worse than the better of the individual baselines. We believe that the reason is that the two data sets have different feature spaces which prevents simple concatenation of the training data.

4.5 Results

The learning curves of the selection task experiments on WSJ test data are shown in Figure 4.1. The three curves in each plot correspond to different feature sets. Accuracy



(a) Articles (b) Prepositions

Figure 4.1: Accuracy for the selection task on WSJ test data.

improves quickly in the beginning but improvements get smaller as the size of the training data increases. The best results are 87.56% for articles (Han) and 68.25% for prepositions (TetreaultParse). The best accuracy for articles is comparable to the best reported results of 87.70% (Lee, 2004) on this data set.

The learning curves of the correction task experiments on NUCLE test data are shown in Figure 4.2 and 4.3. Each sub-plot shows the curves of three models as described in the last section: ASO trained on NUCLE and Gigaword, the baseline classifier trained on NUCLE, and the baseline classifier trained on Gigaword. For ASO, the x-axis shows the number of target problem training instances. The first observation

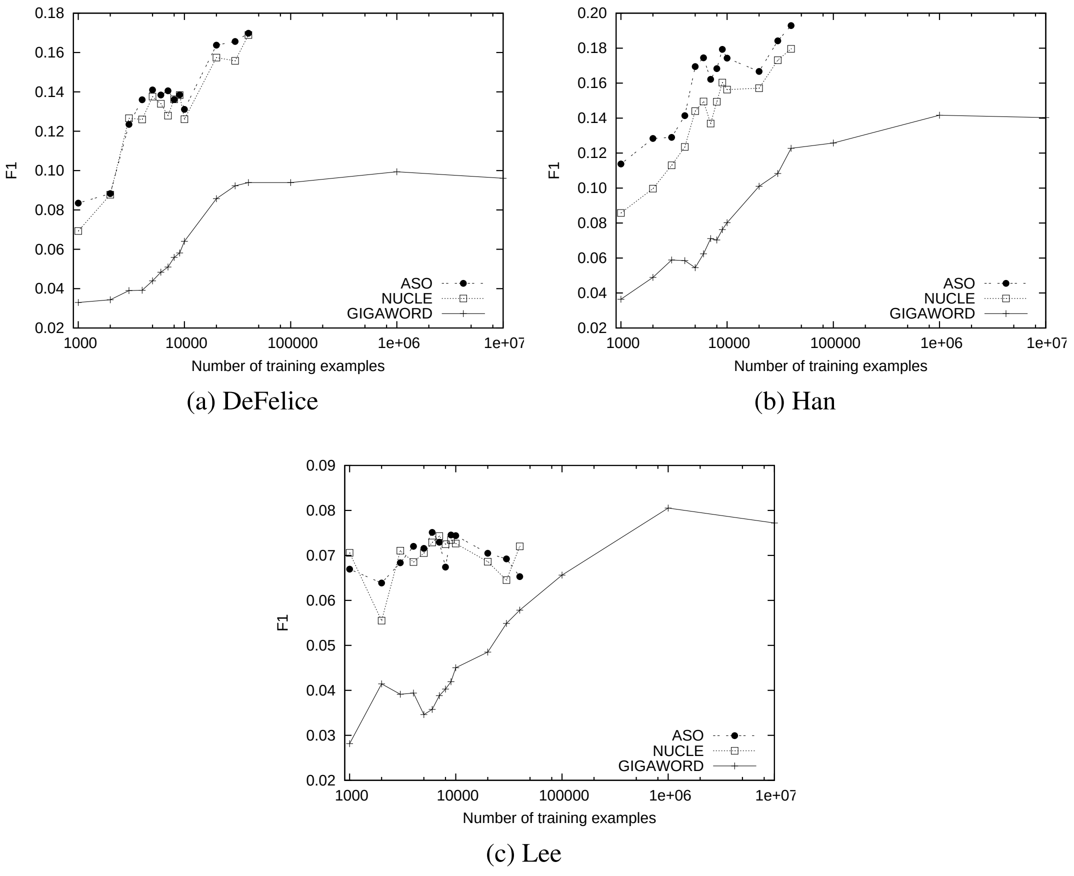
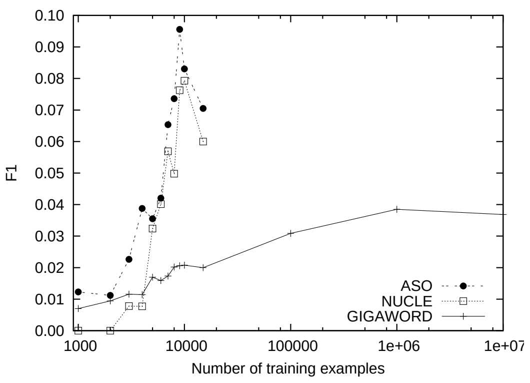
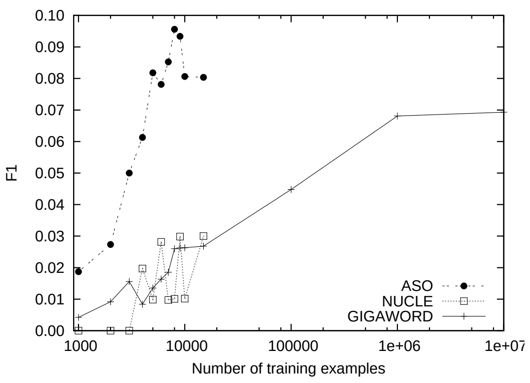


Figure 4.2: F_1 score for the article correction task on NUCLE test data. Each plot shows ASO and two baselines for a particular feature set.

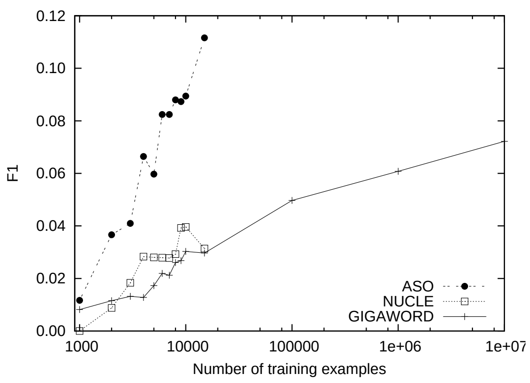
is that high accuracy for the selection task on non-learner text does not automatically entail high F_1 score on learner text. We also note that feature sets with similar performance on non-learner text can show very different performance on learner text. The second observation is that training on annotated learner text can significantly improve performance. In three experiments (articles DeFelice, Han, prepositions DeFelice), the NUCLE model outperforms the Gigaword model trained on 10 million instances. Fi-



(a) DeFelice



(b) TetreaultChunk



(c) TetreaultParse

Figure 4.3: F_1 score for the preposition correction task on NUCLE test data. Each plot shows ASO and two baselines for a particular feature set.

nally, the ASO models show the best results. In the experiments where the NUCLE models already perform better than the Gigaword baseline, ASO gives comparable or slightly better results (articles DeFelice, Han, Lee, prepositions DeFelice). In those experiments where neither baseline shows good performance (TetreaultChunk, TetreaultParse), ASO results in a large improvement over either baseline. The best results are 19.29% F_1 score for articles (Han) and 11.15% F_1 score for prepositions (TetreaultParse) achieved by the ASO model. The model that was trained on the concatenated data from NUCLE and Gigaword achieved no improvement over the NUCLE and Gigaword baselines. For articles, the F_1 score was 14.32% using the Han feature set, and for prepositions, the F_1 score was 1.06% (TetreaultParse) and 2.10% (DeFelice).

4.6 Analysis

In this section, we analyze the results in more detail and show examples from the test set for illustration.

Table 4.1 shows precision, recall, and F_1 score for the best models in our experiments. ASO achieves a higher F_1 score than the Gigaword and NUCLE baselines. We use the sign-test with bootstrap re-sampling for statistical significance testing. The sign-test is a non-parametric test that makes fewer assumptions than parametric tests like the t-test. The improvements in F_1 score of ASO over either baseline are statistically significant ($p < 0.01$) for both articles and prepositions.

Articles			
Model	**Prec**	**Rec**	**F_1**
Gigaword (Han)	10.33	**21.81**	14.02
NUCLE (Han)	**29.48**	12.91	17.96
NUCLE+Gigaword (Han)	11.24	19.72	14.32
ASO (Han)	26.44	15.18	**19.29**

Prepositions			
Model	**Prec**	**Rec**	**F_1**
Gigaword (TetreaultParse)	4.77	**14.81**	7.21
NUCLE (DeFelice)	13.84	5.55	7.92
NUCLE+Gigaword (TetreaultParse)	3.84	0.62	1.06
NUCLE+Gigaword (DeFelice)	1.47	3.70	2.10
ASO (TetreaultParse)	**18.30**	8.02	**11.15**

Table 4.1: Best results for the correction task on NUCLE test data. Improvements for ASO over either baseline are statistically significant ($p < 0.01$) for both tasks.

The difficulty in grammatical error correction is that in many cases, more than one word choice can be correct. Even with a threshold, the Gigaword baseline model suggests too many corrections, because the model cannot make use of the observed word as a feature. This results in low precision. For example, the model replaces *as* with *by* in the sentence *This group should be categorized* ***as*** *the vulnerable group*, which is wrong. In contrast, the NUCLE model learns a bias towards the observed word and therefore achieves higher precision. However, the training data is smaller and therefore recall is low as the model has not seen enough examples during training. This is especially true for prepositions which can occur in a large variety of contexts. For example, the preposition *in* should be *on* in the sentence ... *psychology had an impact* ***in*** *the way we process and manage technology*. The phrase *impact on the way* does not appear in the NUCLE training data and the NUCLE baseline fails to detect the error. The ASO model is able to take advantage of both the annotated learner text and the large non-learner text, thus achieving overall higher F_1 score. The phrase *impact on the way*, for example, appears many times in the Gigaword training data. With the common structure learned from the auxiliary problems, the ASO model successfully finds and corrects this mistake.

4.6.1 Manual Evaluation

Articles			
	ASO	**System A**	**System B**
(1) Correct	4	1	1
(2) Both Ok	16	12	18
(3) Both Wrong	0	1	0
(4) Other Error	1	0	0
(5) False Flag	1	0	4
(6) Miss	3	5	6
(7) No Flag	975	981	971
Precision	**80.00**	50.00	20.00
Recall	**57.14**	14.28	14.28
F_1	**66.67**	22.21	16.67

Prepositions			
	ASO	**System A**	**System B**
(1) Correct	3	3	0
(2) Both Ok	35	39	24
(3) Both Wrong	0	2	0
(4) Other Error	0	0	0
(5) False Flag	5	11	1
(6) Miss	12	11	15
(7) No Flag	1945	1934	1960
Precision	**37.50**	18.75	0.00
Recall	**20.00**	18.75	0.00
F_1	**26.09**	18.75	0.00

Table 4.2: Manual evaluation and comparison with commercial grammar checking software.

We carried out a manual evaluation of the best ASO models and compared their output with two commercial grammar checking software packages which we call *Sys-*

tem A and *System B*. We randomly sampled 1000 test instances for articles and 2000 test instances for prepositions and manually categorized each test instance into one of the following categories: *(1) Correct* means that both human and system flag an error and suggest the same correction. If the system's correction differs from the human but is equally acceptable, it is considered *(2) Both Ok*. If the system identifies an error but fails to correct it, we consider it *(3) Both Wrong*, as both the writer and the system are wrong. *(4) Other Error* means that the system's correction does not result in a grammatical sentence because of another grammatical error that is outside the scope of article or preposition errors, e.g., a noun number error as in *"all **the** dog"*. If the system corrupts a previously correct sentence it is a *(5) False Flag*. If the human flags an error but the system does not, it is a *(6) Miss*. *(7) No Flag* means that neither the human annotator nor the system flags an error. We calculate precision by dividing the count of category (1) by the sum of counts of categories (1), (3), and (5), and recall by dividing the count of category (1) by the sum of counts of categories (1), (3), and (6). The results are shown in Table 4.2. The ASO method outperforms both commercial software packages. This evaluation shows that even commercial software packages achieve low F_1 score for article and preposition errors, which confirms the difficulty of these tasks.

4.7 Conclusion

In this chapter, we have presented a novel approach to grammatical error correction based on Alternating Structure Optimization. Our experiments for article and preposition errors show the advantage of the ASO approach over two baseline methods. The ASO approach also outperforms two commercial grammar checking software packages in a manual evaluation.

Chapter 5

Lexical Choice Errors

The de facto standard approach to grammatical error correction is to build a statistical classification model that can pick the most likely correction from a *confusion set* of possible correction choices. The way the confusion set is defined depends on the type of error. For articles and preposition errors, which have attracted the most attention in grammatical error correction, the confusion sets are based on the syntactic part of speech of the word. Work in context-sensitive spelling error correction (Golding and Roth, 1999) has traditionally focused on confusion sets with similar spelling ({*dessert*, *desert*}) or similar pronunciation ({*there*, *their*}). In other words, the words in a confusion set are deemed confusable because of orthographic or phonetic similarity.

In contrast, in this chapter, we investigate a class of grammatical errors where the source of confusion is the similar *semantics* of the words, rather than orthography, phonetics, or syntax. In particular, we focus on *lexical choice errors* and *collocation errors* in EFL writing. The term *collocation* (Firth, 1957) describes a sequence of words that is conventionally used together in a particular way by native speakers and appears more often together than one would expect by chance. The correct use of collocations is a major difficulty for EFL students (Farghal and Obiedat, 1995). For the remainder of this chapter, we will use the terms lexical choice error and collocation error interchangeable.

In this chapter, we present a novel approach for automatic correction of lexical choice errors in EFL writing. Our key observation is that words are potentially confusable for an EFL student if they have similar translations in the writer's first language (L1-language), or in other words if they have the same semantics in the L1-language of the writer. The Chinese word 看 (*kàn*), for example, has over a dozen translations in English, including the words *see*, *look*, *read*, and *watch*. A Chinese speaker who still "thinks" in Chinese has to choose from all these possible translations when he wants to express a sentence like *I like to* ***watch*** *movies* and might instead produce a sentence like **I like to* ***look*** *movies*. Although the semantics of *watch* and *look* are similar, the former is clearly the more fluent choice in this context. While these types of *L1-transfer errors* have been known in the EFL teaching literature (Swan and Smith, 2001; Meng, 2008), research in grammatical error correction has mostly ignored this fact. These results were first published in (Dahlmeier and Ng, 2011a).

We first analyze lexical choice errors in the NUS Corpus of Learner English (NUCLE). Our analysis confirms that many lexical choice errors can be traced to similar translations in the writer's L1-language. Based on this result, we propose a novel approach for automatic lexical choice error correction. The key component in our approach are *L1-induced paraphrases* which we automatically extract from an L1-English parallel corpus. Our proposed approach outperforms traditional approaches based on edit distance, homophones, and WordNet synonyms on a test set of real-world learner data in an automatic and a human evaluation. Finally, we present a detailed analysis of unsolved instances in the data set to highlight possible directions for future work.

This work adds to a growing body of research that leverages parallel corpora for semantic NLP tasks, for example in word sense disambiguation (Ng et al., 2003; Chan and Ng, 2005; Ng and Chan, 2007; Zhong and Ng, 2009), paraphrasing (Bannard and Callison-Burch, 2005; Liu et al., 2010a), and machine translation evaluation (Snover et al., 2009; Liu et al., 2010b).

The remainder of this chapter is organized as follows. Section 5.1 presents our analysis of lexical choice errors. Section 5.2 describes our approach for automatic lexical choice error correction. The experimental setup and the results are described in Sections 5.3 and 5.4, respectively. Section 5.5 provides further analysis. Section 5.6

Word tokens	1,220,257
Total number of errors	46,597
Lexical choice errors	2,747
Unique lexical choice errors	2,412
Avg. lexical choice error length (words)	1.17
Avg. correction length (words)	1.13

Table 5.1: Lexical errors statistics of the NUCLE corpus

concludes the chapter.

5.1 Analysis of EFL Lexical Choice Errors

While the fact that lexical choice errors can be caused by L1-transfer has been ascertained by EFL researchers (Meng, 2008), we need to quantify how frequent lexical choice errors can be traced to these types of transfer errors in order to estimate how many errors in EFL writing one can potentially hope to correct with information about the writer's L1-language.

We base our analysis on the NUCLE corpus. The details of the corpus are described in Chapter 3. Some statistics of the corpus with respect to lexical choice errors are summarized in Table 5.1. Most of the students are native Chinese speakers. In this chapter, we focus on errors which have been marked with the error tag *wrong collocation/idiom/preposition*. As preposition errors are not the focus of this chapter, we automatically filter out all instances which represent simple substitutions of prepositions, using a fixed list of frequent English prepositions. In a similar way, we filter out a small number of article errors which were marked as collocation errors. Finally, we filter out instances where the annotated phrase or the suggested correction is longer than 3 words, as we observe that they contain highly context specific corrections and are unlikely to generalize well (e.g., *for the simple reasons that these can help them* → *simply to*). After filtering, we end up with 2,747 lexical choice errors and their respective corrections, which account for about 6% of all errors in the corpus. This makes lexical choice errors the 7th largest class of errors in the corpus. Not counting duplicates, there are 2,412 distinct lexical choice errors and corrections. Lexical choice errors represent a particular challenge as the possible corrections are not restricted to a closed set of choices and they are directly related to *semantics* rather than syntax. We analyzed the lexical choice errors and found that they can be attributed to the following sources of confusion:

Spelling: We suspect that an error is caused by similar orthography if the edit distance between the erroneous phrase and its correction is less than a certain threshold.

Homophones: We suspect that an error is caused by similar pronunciation if the erroneous word and its correction have the same pronunciation. We use the CuVPlus English dictionary (Mitton, 1992) to map words to their phonetic representations.

Synonyms: We suspect that an error is caused by synonymy if the erroneous word and its correction are synonyms in WordNet (Fellbaum, 1998). We use WordNet version 3.0.

L1-transfer: We suspect that an error is caused by L1-transfer if the erroneous phrase and its correction share a common translation in a Chinese-English phrase table. The details of the phrase table construction are described in Section 5.2.1. We note that although we focus on Chinese-English translations, this method is applicable to any language pair where parallel corpora are available.

As CuVPlus and WordNet are defined for individual words, we extend the matching process to phrases in the following way: two phrases A and B are deemed homophones/synonyms if they have the same length and the i-th word in phrase A is a homophone/synonym of the corresponding i-th word in phrase B.

The results of the analysis are shown in Table 5.2. Tokens refer to running erroneous phrase-correction pairs including duplicates and types refers to distinct erroneous phrase-correction pairs. As a lexical choice error can be part of more than one category, the rows in the table do not sum up to the total number of errors. The number of errors that can be traced to L1-transfer greatly outnumbers all other categories. The table also shows the number of lexical choice errors that can be traced to L1-transfer but not the other sources. 906 lexical errors with 692 distinct lexical choice error types can be attributed only to L1-transfer but not to spelling, homophones, or synonyms. Table 5.3

Suspected Error Source	**Tokens**	**Types**
Spelling	154	131
Homophones	2	2
Synonyms	74	60
L1-transfer	1016	782
L1-transfer without spelling	954	727
L1-transfer without homophones	1015	781
L1-transfer without synonyms	958	737
L1-transfer without spelling, homophones, synonyms	906	692

Table 5.2: Analysis of lexical errors. The threshold for spelling errors is one for phrases of up to six characters and two for the remaining phrases.

Spelling	. . . it received *critics (criticism)* as much as complaints . . .
	. . . budget for the aged to *improvise (improve)* other areas.
Homophones	. . . diverse spending can *aide (aid)* our country.
	. . . *insure (ensure)* the safety of civilians . . .
Synonyms	. . . rapid *increment (increase)* of the seniors . . .
	. . . energy that we can *apply (use)* in the future . . .
L1-transfer	. . . and *give (provide,* 给予 *)* reasonable fares to the public . . .
	. . . and *concerns (attention,* 关注 *)* that the nation put on technology and engineering . . .

Table 5.3: Examples of lexical choice errors with different sources of confusion. The correction is shown in parenthesis. For L1-transfer, we also show an example of a shared Chinese translation. The L1-transfer examples shown here do not belong to any of the other categories.

shows some examples of lexical choice errors for each category from the corpus. We note that there are also lexical choice error types that cannot be traced to any of the above sources. We will return to these errors in Section 5.5.

5.2 Correcting Lexical Choice Errors

Once we have identified the confusion sets, we could use standard techniques from context-sensitive spelling correction, like linear classifiers, to choose the correct word from a confusion set during test time. However, this has the drawback that we would have to train a separate classifier for each confusion set which is impractical in our case. Furthermore, words can appear in multiple confusion sets which would require an additional step for picking the right confusion set during testing. In this section, we propose an alternative approach for correcting lexical choice errors in EFL writing based on paraphrasing.

5.2.1 L1-induced Paraphrases

We use the popular technique of paraphrasing with parallel corpora (Bannard and Callison-Burch, 2005) to automatically find lexical choice candidates from a sentence-aligned L1-English parallel corpus. As most of the essays in the NUCLE corpus are written by native Chinese speakers, we use the FBIS Chinese-English corpus, which consists of about 230,000 Chinese sentences (8.5 million words) from news articles, each with a single English translation. We tokenize and lowercase the English half of the corpus in the standard way. We segment the Chinese half of the corpus using the maximum entropy segmenter from (Low et al., 2005). Subsequently, we automatically align the texts at the word level using the Berkeley aligner (Liang et al., 2006; Haghighi et al., 2009). We extract English-L1 and L1-English phrases of up to three words from the aligned texts using the widely used phrase extraction heuristic in (Koehn et al., 2003). The paraphrase probability of an English phrase e_1 given an English phrase e_2 is defined as

$$p(e_1|e_2) = \sum_f p(e_1|f)p(f|e_2) \tag{5.1}$$

where f denotes a foreign phrase in the L1 language. The phrase translation probabilities $p(e_1|f)$ and $p(f|e_2)$ are estimated by maximum likelihood estimation and smoothed using Good-Turing smoothing (Foster et al., 2006). Finally, we only keep paraphrases with a probability above a certain threshold (set to 0.001 in this work).

5.2.2 Lexical Choice Correction with Phrase-based SMT

We implement the approach in the framework of phrase-based statistical machine translation (SMT) (Koehn et al., 2003). Phrase-based SMT tries to find the highest scoring translation $\mathbf{e}$ given an input sentence $\mathbf{f}$. The *decoding* process of finding the highest scoring translation is guided by a log-linear model which scores translation candidates using a set of feature functions $h_i,\ i = 1, \ldots, n$

$$score(\mathbf{e}|\mathbf{f}) = \exp\left(\sum_{i=1}^{n} \lambda_i h_i(\mathbf{e}, \mathbf{f})\right) . \tag{5.2}$$

Typical features include a phrase translation probability $p(e|f)$, an inverse phrase translation probability $p(f|e)$, a language model score $p(\mathbf{e})$, and a constant phrase penalty. The optimization of the feature weights $\lambda_i,\ i = 1, \ldots, n$ can be done using minimum error rate training (MERT) (Och, 2003) on a development set of input sentences and their reference translations.

Because of the great flexibility of the log-linear model, researchers have used the framework for other tasks outside SMT, including grammatical error correction (Brockett et al., 2006). We adopt a similar approach in this work. We modify the phrase table of the popular phrase-based SMT decoder MOSES (Koehn et al., 2007) to include lexical choice corrections with features derived from spelling, homophones, synonyms, and L1-induced paraphrases.

- **Spelling:** For each English word, the phrase table contains entries consisting of the word itself and each word that is within a certain edit distance from the original word. Each entry has a constant feature of 1.0.

- **Homophones:** For each English word, the phrase table contains entries consisting of the word itself and each of the word's homophones. We determine homophones using the CuVPlus dictionary. Each entry has a constant feature of 1.0.

- **Synonyms:** For each English word, the phrase table contains entries consisting of the word itself and each of its synonyms in WordNet. If a word has more than one sense, we consider all its senses. Each entry has a constant feature of 1.0.

- **L1-paraphrases:** For each English phrase, the phrase table contains entries consisting of the phrase and each of its L1-derived paraphrases as described in Section 5.2.1. Each entry has two real-valued features: a paraphrase probability according to Equation 5.1 and an inverse paraphrase probability.

- **Baseline** We combine the phrase tables built for spelling, homophones, and synonyms. The combined phrase table contains three binary features for spelling, homophones, and synonyms, respectively.

- **All** We combine the phrase tables from spelling, homophones, synonyms and L1-paraphrases. The combined phrase table contains five features: three binary features for spelling, homophones, and synonyms and two real-valued features for the L1-paraphrase probability and inverse L1-paraphrase probability.

Additionally, each phrase table contains the standard constant phrase penalty feature. The first four tables only contain lexical choice candidates for individual words. We leave it to the decoder to construct corrections for longer phrases during the decoding process if necessary.

5.3 Experiments

In this section, we empirically evaluate our approach on real lexical choice errors in learner English.

5.3.1 Data Set

We randomly sample a development set of 770 sentences and a test set of 856 sentences from the NUCLE corpus. Each sentence contains exactly one lexical choice error. The sampling is performed in a way that sentences from the same document cannot end up in both the development and the test set. In order to keep conditions as realistic as possible, we make no attempt to filter the test set in any way.

We build phrase tables as described in Section 5.2.2. In practice, we only need to generate phrase table entries for words and phrases which actually appear in the development or test set. Paraphrases are extracted from the FBIS Chinese-English corpus.

For the language model, we use a 5-gram language model trained on the English Gigaword corpus[9] with modified Kneser-Ney smoothing.

5.3.2 Evaluation Metrics

We conduct an automatic and a human evaluation. The main evaluation metric is *mean reciprocal rank (MRR)* which is the arithmetic mean of the inverse ranks of the first correct answer returned by the system

$$\text{MRR} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank(i)} \tag{5.3}$$

where N is the size of the test set. If the system did not return a correct answer for a test instance we set $\frac{1}{rank(i)}$ to zero. In the human evaluation, we additionally report precision at rank $k,\ k = 1, 2, 3$ which we calculate as follows

$$\text{P@k} = \frac{\sum_{a \in A} score(a)}{|A|} \tag{5.4}$$

where A is the set of returned answers of rank k or less and $score(\cdot)$ is a real-valued scoring function between zero and one.

5.3.3 Lexical Choice Error Experiments

Automatic correction of lexical choice errors can conceptually be divided into two steps: *i) identification* of wrong lexical choices in the input, and *ii) correction* of the identified lexical errors. In this work, we focus on the second step and assume that the erroneous lexical choice has already been identified. While this might seem like a simplification, it has been the common evaluation setup in lexical choice error correction (see for example (Wu et al., 2010)). It also has a practical application where the user first selects a word or phrase and the system displays possible corrections.

In our experiments, we use the start and end offset of the lexical error provided

[9] LDC2009T13

by the human annotator to identify the location of the lexical choice error. We fix the translation of the rest of the sentence to its identity. We remove phrase table entries where the source phrase and target phrase are identical, thus practically forcing the system to change the identified phrase. We set the distortion limit of the decoder to zero to achieve monotone decoding. We previously observed that word order errors are virtually absent in lexical choice errors in the NUCLE corpus. All experiments use the same 5-gram language model from the Gigaword corpus to allow a fair comparison. We perform MERT training with the popular BLEU metric (Papineni et al., 2002) on the development set of erroneous sentences and their corrections. As the search space is restricted to changing a single phrase per sentence, training converges relatively quickly after two or three iterations. After convergence, the model can be used to automatically correct new lexical choice errors.

5.4 Results

We evaluate the performance of the proposed method on the test set of 856 sentences, each with one lexical choice error. We conduct both an automatic and a human evaluation. In the automatic evaluation, the system's performance is measured by computing the rank of the gold answer provided by the human annotator in the N-best list of the system. We limit the size of the N-best list to the top 100 outputs. If the gold answer is not found in the top 100 outputs, the rank is considered to be infinity, or in other words the inverse of the rank is zero. We also report the number of test instances for which the gold answer was ranked among the top $k,\ k = 1, 2, 3, 10, 100$ answers. The results of the automatic evaluation are shown in Table 5.4

For lexical choice errors, there are usually more than one possible correct answer. Therefore the automatic evaluation underestimates the actual performance of the system by only considering the single gold answer as correct and all other answers as wrong. Therefore, we carried out a human evaluation for the systems BASELINE and ALL. We recruited two English speakers to judge a subset of 500 test sentences. For each sen-

Model	**Rank = 1**	**Rank ≤ 2**	**Rank ≤ 3**	**Rank ≤ 10**	**Rank ≤ 100**	**MRR**
Spelling	35	41	42	44	44	4.51
Homophones	1	1	1	1	1	0.11
Synonyms	32	47	52	60	61	4.98
Baseline	49	68	80	93	96	7.61
L1-paraphrases	93	133	154	**216**	**243**	15.43
All	**112**	**150**	**166**	**216**	241	**17.21**

Table 5.4: Results of automatic evaluation. Columns two to six show the number of gold answers that are ranked within the top k answers. The last column shows the mean reciprocal rank in percentage. Bigger values are better.

P(A)	0.8076
Kappa	0.6152

Table 5.5: Inter-annotator agreement. $P(E) = 0.5$.

tence, a judge was shown the original sentence and the 3-best candidates of each of the two systems. We restricted the human evaluation to the 3-best candidates, as we believe that answers at a rank larger than three will not be very useful in a practical application. The candidates are displayed together in alphabetical order without any information about their rank or which system produced them or the gold answer by the annotator. The difference between the candidates and the original sentence is highlighted. The judges were asked to make a binary judgment for each of the candidates on whether the proposed candidate is a valid correction of the original or not. We represent valid corrections with a score of 1.0 and non-valid corrections with a score of 0.0. Inter-annotator agreement is reported in Table 5.5. The probability of agreement $P(A)$ is the percentage of times that the annotators agree, and $P(E)$ is the expected agreement by chance, which is 0.5 in this case. We obtain a Kappa coefficient of 0.6152. A Kappa coefficient between 0.6 and 0.8 is considered as showing *substantial* agreement according to Landis and Koch (1977). To compute precision at rank k, we average the judgments. Thus, a system can receive a score of 0.0 (both judgments negative), 0.5 (judges disagree), or 1.0 (both judgments positive) for each returned answer. To compute MRR, we cannot simply average the judgments as MRR requires binary judgments whether an item is correct or not. Instead, we report MRR on the union and the intersection of the judgments. In the first case, the rank of the first correct item is the minimum rank of any item judged correct by *either* judge. In the second case, the rank of the first correct

Model	Rank = 1	Rank ≤ 2	Rank ≤ 3	P@1	P@2	P@3	MRR
Baseline	43 | 141	69 | 201	83 | 237	18.40	16.68	15.36	12.13 | 36.60
All	**137 | 245**	**176 | 303**	**204 | 340**	**38.20**	**32.87**	**29.30**	**33.16 | 57.26**

Table 5.6: Results of human evaluation. Rank and MRR results are shown for the intersection (first value) and union (second value) of human judgments.

item is the minimum rank of any item judged correct by *both* judges. The results for the human evaluation are shown in Table 5.6. The best system ALL outperforms the BASELINE approach on all measures. It receives a precision at rank 1 of 38.20% and a MRR of 33.16% (intersection) and 57.26% (union). Table 5.7 shows examples from the test set.

Unfortunately, comparison of our results with previous work is complicated by the fact that there currently exists no standard data set for lexical choice error correction. With the NUCLE corpus now publicly available, we hope that it will allow researchers to more directly compare their results in future.

5.5 Analysis

In this section, we analyze and categorize those test instances for which the ALL system could not produce an acceptable correction in the top 3 candidates. We manually analyze 100 test sentences for which neither judge had deemed any candidate answer to be a valid correction. Based on our findings, we categorize the 100 sentences into eight categories which are shown below. Table 5.8 shows examples from each category.

Out-of-vocabulary (21/100) The most frequent reason why the system does not produce a good correction is that the erroneous lexical choice is out of vocabulary. These lexical choice errors often involve compound words, like *man-hours* or *carefully-nurturing*, or infrequent expressions, like *copy phenomena*, which do not appear in the FBIS parallel corpus. We expect that this problem can be reduced by using larger parallel corpora for paraphrase extraction.

Near miss (18/100) The second largest category consists of instances where the system barely misses the gold standard answer. This includes cases where the extracted L1-

Original	it must be clear, concise and unambiguous to *prevent* any off-track
Gold	it must be clear, concise and unambiguous to *avoid* any off-track
All	it must be clear, concise and unambiguous to *avoid* any off-track
	it must be clear, concise and unambiguous to *stop* any off-track
	it must be clear, concise and unambiguous to *block* any off-track
Baseline	*it must be clear, concise and unambiguous to *present* any off-track
	it must be clear, concise and unambiguous to *forestall* any off-track
	*it must be clear, concise and unambiguous to *lock* any off-track

Original	although many may *agree* that public spending on the eldery should be limited . . .
Gold	although many may *argue* that public spending on the eldery should be limited . . .
All	although many may *believe* that public spending on the eldery should be limited . . .
	although many may *think* that public spending on the eldery should be limited . . .
	although many may *accept* that public spending on the eldery should be limited . . .
Baseline	*although many may *agreed* that public spending on the eldery should be limited . . .
	*although many may *hold* that public spending on the eldery should be limited . . .
	*although many may *agrees* that public spending on the eldery should be limited . . .

Table 5.7: Examples of test sentences with the top 3 answers of the ALL and BASELINE system. An answer judged incorrect by at least one judge is marked with an asterisk (*).

Out of vocabulary	… many illegal *copy phenomena (copy phenomena, **copies**)* in china. … lead to reduced *man-hours (man-hours, **productivity**)* as people fall sick …
Near miss	… smaller groups of people, sometimes *even (more, **only**)* individual . … take pre-emptive *actions (activities, **measures**)* …
Function/auxiliary words	… entertainment an eldery person can *have (be, **enjoy**)* . … and the security issue is solved *also (and, **too**)*
Discourse specific	… make other countries respect and fear *you (<question mark>, **a country**)* … will contribute nothing to the *accident (explosion, **problem**)* .
Spelling errors	this *incidence (rate, **incident**)* had also resulted in 4 fatalities … refrigerator did not *compromise (yield, **comprise**)* of any moving parts …
Word sense	… refers to the desire or shortage of a *good (better, **commodity**)* and … … members are always from different *majors (major league, **specialities**)*
Preposition	… can be an area worth *investing (investing, **investing in**)* … in spending their *resources (resources, **resources on**)*
Others	this might *redirect (make sound, **reduce**)* foreign investments … … a trading hub since *british 's (british 's, **british**)* rule.

Table 5.8: Examples of sentences without valid corrections by the ALL model. The top 1 suggestion of the system and the gold answer (in bold) are shown in parenthesis.

paraphrases do not contain the exact phrase required, e.g., the paraphrase table contains *even* → *only get* when the gold correction is *even* → *only*, or the phrase table actually contains the gold answer but fails to rank it among the top 3 answers. The first problem could be addressed by modifying the phrase extraction heuristic to produce more fine-grained phrase pairs. The second problem requires a better language model. Although the language model is trained on the large English Gigaword corpus, it is not always successful in promoting the correct candidate to the top. The domain mismatch between the newswire domain of Gigaword and student essays could be one reason for this.

Function/auxiliary words (14/100) We observe that lexical choice errors that involve function words or auxiliary words are not handled very well by our model. Function words and auxiliary words in English lack direct counterparts in Chinese, which is why the word alignments and therefore the extracted phrases for these words contain a high amount of noise. As function words and auxiliaries are essentially a closed set, it might be more promising to build separate models with fixed confusion sets for them.

Discourse specific (14/100) Some of the gold answers are highly specific to the particular discourse that they appear in. As our model corrects lexical choice errors at the sentence level, such gold answers will be very difficult or impossible to determine correctly. Including more context beyond the sentence level might help to overcome this problem, although it is not easy to integrate this larger context information.

Spelling errors (9/100) Some of the lexical choice errors are caused by spelling mistakes, e.g., *incidence* instead of *incident*. Although the ALL model includes candidates which are created through edit distance, paraphrase candidates created from the misspelled word can dominate the top 3 ranks, e.g. *rate* and *frequently* are paraphrases of *incidence*. A possible solution would be to perform spell-checking as a separate pre-processing step prior to lexical choice correction.

Word sense (7/100) Some of the failures of the model can be attributed to ambiguous senses of the source phrase. As we do not perform word sense disambiguation, candidates from other word senses can end up as the top candidates. Including word sense disambiguation into the model might be able to help, although accurate word sense disambiguation on noisy learner text is not easy.

Preposition (6/100) Some of the lexical choice errors involve preposition constructions, e.g., the student wrote *attend* instead of *attend to*. Because prepositions do not have a direct counterpart in Chinese, the L1-paraphrases do not model their semantics very well. This category is closely related to the function/auxiliary word category. Again, since prepositions are a closed set, it might be more promising to built a separate model for them, like the one described in the last chapter.

Others (11/100) Other mistakes include lexical choice errors where the gold answer slightly changed the semantics of the target word, e.g., *redirect potential foreign investments* → *reduce potential foreign investments*, active-passive alternation (*enhanced economics* → *was economical*), and noun possessive errors (*british 's rule* → *british rule*).

5.6 Conclusion

In this chapter, we have presented a novel approach for correcting lexical choice errors in written learner text. Our approach exploits the semantic similarity of words in the writer's L1-language based on paraphrases extracted from an L1-English parallel corpus. Experiments on real-world learner data show that our approach outperforms traditional approaches based on edit distance, homophones, and synonyms by a large margin.

Chapter 6

A Pipeline Architecture for Grammatical Error Correction

The previous chapters have presented grammatical error correction approaches such that each chapter focuses on a particular error category. An error correction system that can only correct one type of error will be of limited use to a language learner in a practical application. Instead, a practical error correction system has to be able to correct the various types of errors that language learners make. Thus, development and evaluation of more comprehensive end-to-end error correction systems is an important step towards building a practical error correction system for language learners.

In this chapter, we present a pipeline architecture for end-to-end grammatical error correction systems. The idea is simple: the system consists of a pipeline of sequential correction steps. Each step corrects a single error type. The correction algorithm can be a learning-based classifier or a rule-based approach. The proposed corrections after every step are filtered using a language model and only corrections that strictly increase the language model score are kept. The output of one step serves as the input to the next step. The output of the last correction step is the final output of the system.

The architecture is evaluated in the context of the two Helping Our Own (HOO) shared tasks which were the first public evaluation campaigns for end-to-end grammatical error correction systems. Our group from NUS participated in both shared tasks with systems that used the pipeline architecture described above. The NUS submission achieved the second highest correction F_1 score in the HOO 2011 shared task and the highest correction F_1 score in the HOO 2012 shared task. The results were published as part of the proceedings of the HOO shared tasks in (Dahlmeier et al., 2011) and (Dahlmeier et al., 2012), respectively.

The remainder of this chapter is organized as follows. The next section describes the two HOO shared tasks. Section 6.2 describes the architecture of the systems in detail. Section 6.3 describes the features used. Section 6.4 presents the experimental setup and Section 6.5 the results. Section 6.6 provides further discussion. Section 6.7 concludes the chapter.

6.1 The HOO Shared Tasks

HOO 2011

The Helping Our Own 2011 (HOO 2011) (Dale and Kilgarriff, 2011)[10] shared task was the first shared task for grammatical error correction. The organizers of the shared task, Robert Dale and Adam Kilgariff, were motivated by the fact that for NLP researchers, language is both the subject of their scientific work as well as a medium to communicate their research. Scientific proceedings are routinely published in English but many researchers are not native English speakers. Writing fluent English papers can be challenging for them. At the same time, the NLP community has developed mature algorithms to analyze English text. The natural conclusion for the organizers was to suggest a shared task that would utilize this know-how to develop authoring tools that help our own researchers to write better papers. That is why the shared task was called "Helping Our Own". In the shared task, participants were provided with a set of documents extracted from papers written by non-native speakers of English. The task was to automatically detect and correct *any* type of grammatical error. The full set of error categories included in the task can be found in (Nicholls, 2003). Participants could ei-

[10]The first HOO shared task was simply called "Helping Our Own (HOO)". To avoid confusion with the HOO 2012 shared task, we refer to the HOO shared task in 2011 as HOO 2011.

ther submit the corrected text produced by their system or a set of corrections (called *edits*).

HOO 2012

The HOO 2012 shared task (Dale et al., 2012) changed the tasks from correcting the writing of NLP researchers to the more general task of correcting the writing of EFL students. The organizers further decided to restrict the task from correcting any type of error to the correction of determiner and preposition errors, which are the most frequent errors made by EFL students. In the HOO 2012 shared task, participants had to submit a set of corrections. The submission of plain text output was not allowed. Evaluation in both HOO shared tasks was done by computing precision, recall, and F_1 score between the system edits and a manually created set of gold-standard edits (corrections).

6.2 System Architecture

In this section, we describe the pipeline architecture of the NUS systems that participated in the HOO 2011 and HOO 2012 shared tasks. Both systems target spelling, article, and preposition errors in a pipeline of rule and classifier-based correction steps. Each correction step takes one-sentence-per-line plain text as input and outputs a one-sentence-per-line plain text in return. Both systems treat article and preposition correction as classification problems. We use linear classifiers to predict the correct word from a confusion set of possible correction options. Separate classifiers are built for article, preposition replacement, preposition insertion, and preposition deletion errors. Each article or preposition correction step involves three internal steps:

1. Feature extraction
2. Classification
3. Language model filter

Feature extraction first analyzes the syntactic structure of the input sentences (POS tagging, chunking, and parsing) and identifies relevant instances for correction (e.g., all noun phrases (NP) for article correction). Each instance is mapped to a real-valued feature vector. Next, a classifier predicts the most likely correction for each feature vector. Finally, the proposed corrections are filtered using a language model and only corrections that strictly increase the language model score are kept. There is no additional threshold parameter for the language model filter.

HOO 2011

For the HOO 2011 shared task, the NUS system consists of a sequential pipeline of three processing steps:

1. Spelling correction
2. Article correction
3. Replacement preposition correction

HOO 2012

For the HOO 2012 shared task, the pipeline contains an additional step for preposition insertion and deletion correction.

1. Spelling correction
2. Article correction
3. Replacement preposition correction
4. Missing and unwanted preposition correction

The details of each processing step are presented in the remainder of this section.

6.2.1 Pre- and Post-Processing

Sentence segmentation and tokenization are carried out on the input files in a pre-processing step. In the case of the HOO 2012 shared task, we noticed that some documents were written in all upper case. That has a negative effect on tagging and classification accuracy. Therefore, in the HOO 2012 shared task, pre-processing additionally involves case normalization. We automatically identify and re-case upper-case documents using a standard re-casing model from statistical machine translation. Re-casing is modeled as monotone decoding (without reordering) involving translation of an un-cased sentence to a mixed-case sentence. A post-processing step detokenizes the final output after the last step and extracts the edit structures that encode the corrections. The edit structures are the submission to the shared task.

6.2.2 Spelling Correction

The first correction step in the pipeline is spelling correction. Spelling correction is performed first as spelling errors can have a negative effect on tagging and classification accuracy. Although in HOO 2012, spelling errors were not part of the errors that need to be corrected, spelling correction was still performed to improve the accuracy of subsequent correction steps.

We use an open-source spell checker to correct spelling errors. Words are excluded from spelling correction if they are shorter than a threshold (set to 4 characters in this system), or if they include hyphens or upper case characters inside the word. For the HOO 2011 shared task that contains texts from a specialized domain, we use an in-domain spelling dictionary constructed from all words that appear at least ten times in the ACL-ANTHOLOGY data set described in Section 6.4.1. For the HOO 2012 shared task, all words that appear at least ten times in the HOO 2012 training data are added to the standard spelling dictionary. Finally, all spelling corrections proposed by the spell checker are filtered using a language model.

6.2.3 Article Errors

In HOO 2011 and HOO 2012, article errors are part of the larger class of determiner errors. Determiner errors consist of three error types: replacement determiner (RD), missing determiner (MD), and unwanted determiner (UD). In addition to the indefinite and definite article and the null article (*a, an, the, null article* ϵ), determiner errors include other words like demonstratives (*this, that*) and possessives (*my, her*). In practice, however, article errors account for the majority of determiner errors. We therefore focus our efforts on errors involving only articles.

Correction as Classification

We treat article error correction as a multi-class classification problem. A classifier is trained to predict the correct article from a *confusion set* of possible article choices {*a*, *the*, ϵ}, given the sentence context. The article *an* is normalized as *a* and restored later using a rule-based heuristic. During training, every NP in the training data generates one training example. The class $y \in${*a*, *the*, ϵ} is the correct article that the classifiers should learn to predict. When training on learner text, the correct article is either the article annotated by the gold standard or the observed article used by the writer if the article is not annotated (i.e., the article is correct). When training on non-learner text, the text is assumed to be free of grammatical errors and the correct article equals the observed article used by the writer. The surrounding context is represented as a real-valued feature vector $\mathbf{x} \in \mathcal{X}$. The features of the classifiers are described in detail in Section 6.3. If the classifier is trained on non-learner text, the features are only extracted from the surrounding context. The article itself cannot be included in the features as it would be fully predictive of the class. If the classifier is trained on annotated learner text, the observed article can be used as a feature. However, as pointed out in Chapter 4, one challenge in training classifiers for grammatical error correction on learner text is that the data is highly skewed. Training examples without any error (i.e., the observed article equals the correct article) greatly outnumber those examples with an error (i.e., the observed article is different from the correct article). As the observed article is highly correlated with the correct article, the observed article is a valuable feature (Rozovskaya and Roth, 2010b; Dahlmeier and Ng, 2011b). However, the high correlation can have the undesirable effect that the classifier *always* predicts the observed article and never proposes any corrections. To mitigate this problem, we re-sample the training data when training on learner text, either by oversampling examples with an error or undersampling examples without an error. The sampling parameter is chosen through a grid search so as to maximize the F_1 score on the development data. After training, the classifier can be used to predict the correct article for NPs from new unseen sentences.

During testing, every NP in the test data generates one test example. If the article predicted by the classifier differs from the observed article and the difference between the classifier's confidence score for its first choice and the classifier's confidence score for the observed article is higher than some threshold parameter t, the observed article is replaced by the proposed correction. The threshold parameter t is tuned through a grid search so as to maximize the F_1 score on the development data. In the HOO 2011 shared task, we use a single threshold value per classifier. In the HOO 2012 shared task, we use a separate threshold parameter value for each class which we found worked better than using a single threshold value.

Language Model Filter

All corrections are filtered using a large language model. Only corrections that strictly increase the normalized language model score of a sentence are kept. The normalized language model score is defined as

$$score_{lm} = \frac{1}{|\mathrm{s}|} \log Pr(\mathrm{s}), \tag{6.1}$$

where s is the corrected sentence and $|\mathrm{s}|$ is the sentence length in tokens. The final set of article corrections is applied to an input sentence (i.e., replacing the observed article with the predicted article).

6.2.4 Replacement Preposition Correction

Replacement preposition correction (RT) follows the same strategy of multi-class classification and language model filtering as article correction but with a different confusion set and different features. For the HOO 2011 shared task, the confusion set consists of the prepositions *about, among, at, by, for, in, into, of, on, to*, and *with*. For the HOO 2012 shared task, we expanded the confusion set to the 36 prepositions from the preposition classifier presented in Chapter 4[11]. These prepositions account for the majority of preposition replacement errors in the HOO 2012 training data. During training, every prepositional phrase (PP) in the training data which is headed by a preposition from the confusion set generates one training example. The class y is the correct preposition, either as annotated in the gold standard (learner text) or as observed in the text (non-learner text). During testing, every PP in the test data which is headed by a preposition from the confusion set generates one test example. Again, we apply a threshold to bias the classifier towards the observed preposition. For HOO 2011, we use a single threshold value. For HOO 2012, we use a separate threshold parameter value for each preposition. Finally, we filter all corrections with a large language model.

6.2.5 Missing Preposition Correction

For our participating system in HOO 2012, the pipeline further corrects missing and unwanted preposition errors for the seven most frequently missed or wrongly inserted prepositions in the HOO 2012 training data. These preposition are *about, at, for, in, of, on*, and *to*. While developing the system, we found that adding more prepositions did not increase performance in our experiments.

We treat missing preposition (MT) correction as a binary classification problem. Alternatively, missing preposition error correction could be treated as a multi-class problem, but we found that binary classifiers gave better performance in initial experiments. For each preposition p, we train a binary classifier that predicts the presence or absence

[11]*about, along, among, around, as, at, beside, besides, between, by, down, during, except, for, from, in, inside, into, of, off, on, onto, outside, over, through, to, toward, towards, under, underneath, until, up, upon, with, within, without*

of that preposition. Thus, the confusion set consists only of the preposition p and the "null preposition". During training, we require examples of contexts where p should be used and where it should be omitted. As prepositions typically appear before NPs, we take every NP in the training data as one training example. If the preposition p appears right in front of the NP (i.e., the preposition p and the NP form a PP), the example is a positive example, otherwise (i.e., another preposition or no preposition appears before the NP) it is a negative example. During testing, every NP which does not directly follow a preposition generates one test example. If the classifier predicts that the preposition p should have been used in this context with sufficiently high confidence and inserting p increases the normalized language model score, p is inserted before the NP.

6.2.6 Unwanted Preposition Correction

Unwanted preposition correction (UT) is treated as a binary classification problem similar to missing preposition correction but with different training and test examples. When training the classifier for preposition p, every PP where the writer used the preposition p is one training example. If the gold-standard annotation labels p as unwanted, the example is a positive example for deleting p, otherwise it is a negative example. During testing, every PP with the preposition p generates one test example. If the classifier predicts that p should be deleted with sufficiently high confidence and deleting p increases the normalized language model score, p is deleted. We found that separate classifiers for missing and unwanted preposition correction gave slightly better results compared to using a single classifier for both tasks. As the test examples for missing and unwanted preposition correction of a preposition p are disjoint, both steps can be performed in parallel. This also prevents the case of the system "contradicting" itself by first inserting a preposition and later deleting it. We perform missing preposition correction and unwanted preposition correction for each preposition in turn, before moving to the next preposition.

6.2.7 Learning Algorithm

All correction steps except spelling correction use a linear classifier. There exist various learning algorithms to train linear classifiers. In the HOO 2011 task, we use the empirical risk minimization batch algorithm with least square regularization that was used in Chapter 4. This learning algorithm showed good performance compared to a maximum entropy learning algorithm in earlier experiments. In the HOO 2012 shared task, we changed the learning algorithm to confidence-weighted (CW) learning (Dredze et al., 2008; Crammer et al., 2009), which has been shown to perform well for NLP problems with high dimensional and sparse feature spaces. Instead of keeping a single weight vector, CW learning maintains a distribution over weight vectors, parametrized by a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with mean $\boldsymbol{\mu}$ and covariance matrix Σ. In practice, Σ is often approximated by a diagonal matrix (Dredze et al., 2008). CW is an online learning algorithm that proceeds in rounds over a labeled training set $((y_1, \mathbf{x}_1), (y_2, \mathbf{x}_2), \ldots, (y_n, \mathbf{x}_n))$, one example at a time. After the i-th round, CW learning updates the distribution over weight vectors such that the i-th example is predicted correctly with probability at least $0 < \eta < 1$ while choosing the update step that minimizes the Kullback-Leibler (KL) distance from the current distribution. The CW update rule is:

$$\begin{aligned} (\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}) &= \qquad\qquad (6.2) \\ \arg\min_{\boldsymbol{\mu},\Sigma} \; \mathrm{D}_{\mathrm{KL}} & \; (\mathcal{N}(\boldsymbol{\mu}, \Sigma) || \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)) \\ \text{s.t.} & \; Pr[y_i | \mathbf{x}_i, \boldsymbol{\mu}, \Sigma] \geq \eta. \end{aligned}$$

Dredze *et al.* (2008) show that in the binary case, the CW update rule has a closed-form solution. In the multi-class case, there exists no closed-form solution but the solution can be efficiently approximated. We use sequential many-constraint updates (Crammer et al., 2009) and the `Variance` approximation method (Dredze et al., 2008) to solve the update equation. We set the "aggressiveness" parameter ϕ in the learning algorithm to a default value 0.01 and do updates with 2 constraints. We run two epochs over the training set which appears to be sufficient for the classifier to converge. Fine-tuning any of these parameters did not show much effect in initial experiments. Finally, we average the classifiers (Freund and Schapire, 1999) learned during training to form the final classifier which is used for testing. The CW learning algorithm performed slightly better than the empirical risk minimization batch learning algorithm while being significantly faster during training.

6.3 Features

The choice of features can have an important effect on classification performance. For the HOO 2011 shared task, we adopt the features proposed by Han *et al.* (2006) for article correction and the features proposed by Tetreault and Chodorow (2008b) for preposition replacement correction. These features include lexical and POS N-grams, and chunk features which are commonly used features for grammatical error correction. The feature sets were chosen because they had shown good performance in the ASO classification experiments in Chapter 4.

For the HOO 2012 shared task, we extended and improved the feature sets. We added additional features from the HOO 2011 system by Rozovskaya *et al.* (2011) and dependency parse features based on the work of Tetreault *et al.* (2010) who showed that parse features can further increase performance. For all the above features, the observed article or preposition used by the writer is "blanked out" when computing the features. However, we add the observed article or preposition as an additional feature for article and replacement preposition correction. The features described so far are all binary-valued, i.e., they indicate whether some feature is present in the input or not. Another way is to construct real-valued N-gram features by counting the log frequency of surface N-grams on the web or in a web-scale corpus (Bergsma et al., 2009). Web-scale N-gram count features can harness the power of the web in connection with supervised classification and have successfully been used for a number of NLP generation and disambiguation problems (Bergsma et al., 2009; Bergsma et al., 2010). However, we

Feature	Example
Lexical features	
First word in NP†	*black*
Word i before NP ($i = 1, 2$)	{*on, sat*}
Word i after ($i = 1, 2$)†	{*black, door*}
Word after NP	*period*
Bag of words in NP†	{*black, door, mat*}
POS features	
First POS in NP	JJ
POS i before NP ($i = 1, 2$)	{*IN, VBD*}
POS after NP	*period*
Bag of POS in NP	{JJ, NN, NN}
Head word features	
Head of NP†	*mat*
Head POS	NN
Head countable	*yes*

Table 6.1: HOO 2011 features for article correction. Example: "The cat sat on *the black door mat*." † : lexical tokens in lower case.

are not aware of any previous application in grammatical error correction. Web-scale N-gram count features usually use N-grams of consecutive tokens. The release of web-scale parsed corpora like the WaCky project (Baroni et al., 2009) makes it possible to extend the idea to dependency N-grams of child-parent tuples over the dependency arcs in a dependency parse tree, e.g., {(child, node), (node, parent)} for bigrams, {(child's child, child, node), (child, node, parent), (node, parent, parent's parent)} for trigrams. We collect log frequency counts for dependency N-grams from a large dependency-parsed web corpus and use the log frequency count as a feature. We normalize all real-valued feature values to a unit interval $[0, 1]$ to avoid features with larger values dominating features with smaller values.

For the HOO 2011 shared task, the exact features for article correction and replacement preposition correction are listed in Tables 6.1 and 6.2, respectively. For HOO 2012, the features used for article correction, replacement preposition correction, and missing and unwanted preposition correction are listed in Tables 6.3, 6.4, 6.5, and 6.6, respectively. The features were chosen empirically through experiments on the development data.

Feature	Example
Lexical and POS features	
N-grams ($N = 2, 3$)	{*sitting_X, X_the, ..* }
POS N-grams ($N = 2, 3$)	{*VBG_X, X_DT, ..* }
Head word features	
Head of prev VP†	*sitting*
Head of prev NP†	*cat*
Head of next NP†	*mat*
Head prev NP + head next NP†	*cat+mat*
POS head prev NP +POS head next NP	*NN+NN*
Head prev VP + head prev NP + head next NP†	*sitting+cat+mat*
POS head prev VP + POS head prev NP + POS head next NP	*VBG+NN+NN*

Table 6.2: HOO 2011 features for for replacement preposition correction. Example: "He saw a cat sitting *on the mat*." †: lexical tokens in lower case.

6.4 Experiments

In this section, we report experimental results of the pipeline systems for the HOO 2011 and HOO 2012 shared tasks. For each shared task, we report results on two different data sets: a held-out development test split of the released training data and the official test set. In both the HOO 2011 and the HOO 2012 shared task, we only submitted one run of the system for evaluation although participants were allowed to submit multiple runs.

6.4.1 Data Sets

HOO 2011

As we have described in Chapter 3, there were no large, publicly available learner corpora available at the time the HOO 2011 task was announced. Fortunately for the NLP domain, the ACL Anthology[12] and the ACL Anthology Reference Corpus (ARC) (Bird et al., 2008) represented a large, freely available resource for developing algorithms for HOO 2011, even though the data is not annotated with grammatical errors. The HOO 2011 organizers selected a subset of 19 papers from the ACL Anthology which were then annotated for grammatical errors. For system development, we randomly split the files into a tuning set HOO2011-TUNE (9 files) and a held-out development test set

[12]`http://www.aclweb.org/anthology-new`

Feature	**Example**
Lexical features	
Observed article†	*the*
First word in NP†	*black*
Word i before ($i = 1, 2, 3$)†	{*on, sat, ..*}
Word i before NP ($i = 1, 2$)	{*on, sat, ..*}
Word + POS i before ($i = 1, 2, 3$)†	{*on+IN, sat+VBD, ..*}
Word i after ($i = 1, 2, 3$)†	{*black, door, ..*}
Word after NP	*period*
Word + POS i after ($N = 1, 2$)†	{*period+period, ..* }
Bag of words in NP†	{*black, door, mat*}
N-grams ($N = 2, .., 5$)‡	{*on_X, X_black, ..* }
Word before + NP†	*on+black_door_mat*
NP + N-gram after NP ($N = 1, 2, 3$)†	{ *black_door_mat+period, ..*}
Noun compound (NC)†	*door_mat*
Adj + NC†	*black+door_mat*
Adj POS + NC†	*JJ+door_mat*
NP POS + NC†	*JJ_NN_NN+door_mat*
POS features	
First POS in NP	JJ
POS i before ($i = 1, 2, 3$)	{*IN, VBD, ..*}
POS i before NP ($i = 1, 2$)	{*IN, VBD, ..*}
POS i after ($i = 1, 2, 3$)	{*JJ, NN, ..*}
POS after NP	*period*
Bag of POS in NP	{JJ, NN, NN}
POS N-grams ($N = 2, .., 4$)	{*IN_X, X_JJ, ..* }
Head word features	
Head of NP†	*mat*
Head POS	NN
Head word + POS†	*mat+NN*
Head number	*singular*
Head countable	*yes*
NP POS + head†	*JJ_NN_NN+mat*
Word before + head†	*on+mat*
Head + N-gram after NP † ($N = 1, 2, 3$)	*mat+period*, ..
Adjective + head†	*black+mat*
Adjective POS + head†	*JJ+mat*
Word before + adj + head†	*on+black+mat*
Word before + adj POS + head†	*on+JJ+mat*
Word before + NP POS + head†	*on+JJ_NN_NN+mat*
Web N-gram count features	
Web N-gram log counts $N = 3, .., 5$	{log freq(*on a black*), log freq(*on the black*), log freq(*on black*),..}
Dependency features	
Dep NP head-child†	{*mat-black-amod, ..*}
Dep NP head-parent†	*mat-on-pobj*
Dep child-NP head-parent†	{*black-mat-on-amod-pobj, ..*}

Table 6.3: HOO 2012 features for article correction. Example: "The cat sat on *the black door mat*." † : lexical tokens in lower case, ‡: lexical tokens in both original and lower case.

Feature	**Example**
Preposition features	
Prep before + head	*on+mat*
Prep before + NC	*on+door_mat*
Prep before + NP	*on+black_door_mat*
Prep before + adj + head	*on+black+mat*
Prep before + adj POS + head	*on+JJ+mat*
Prep before + adj + NC	*on+black+door_mat*
Prep before + adj POS + NC	*on+JJ+door_mat*
Prep before + NP POS + head	*on+JJ_NN_NN+mat*
Prep before + NP POS + NC	*on+JJ_NN_NN+door_mat*
Verb object features	
Verb obj†	*sat_on*
Verb obj + head†	*sat_on+mat*
Verb obj + NC†	*sat_on+door_mat*
Verb obj + NP†	*sat_on+black_door_mat*
Verb obj + adj + head†	*sat_on+black+mat*
Verb obj + adj POS + head†	*sat_on+JJ+mat*
Verb obj + adj + NC†	*sat_on+black+door_mat*
Verb obj + adj POS + NC†	*sat_on+JJ+door_mat*
Verb obj + NP POS + head†	*sat_on+JJ_NN_NN+mat*
Verb obj + NP POS + NC†	*sat_on+JJ_NN_NN+door_mat*

Table 6.3: (continued)

HOO2011-DEVTEST (10 files). The official HOO 2011 test data (HOO2011-TEST) is completely unobserved during development. We create two training data sets from the ACL Anthology: ACL-ANTHOLOGY includes all non-OCR documents from the Anthology except the 2010 ACL conference and workshop proceedings as these overlap with the HOO data[13]. CL-JOURNAL contains all non-OCR documents from the *Computational Linguistics* journal. In both cases, we filter out section headings, references, tables, etc. The WEB 1T 5-GRAM CORPUS (Brants and Franz, 2006) is used to build the language modeling filter for article and preposition correction. For spelling correction, the language model filter is built from the ACL-ANTHOLOGY data set. In both cases, we build 5-gram language models with Stupid Back-off smoothing (Brants et al., 2007). The linear classifiers for article and preposition correction are trained on the CL-JOURNAL data set. Threshold parameters are tuned on HOO2011-TUNE when testing on HOO2011-DEVTEST, and on the complete HOO 2011 training data when testing on HOO2011-TEST.

[13]Although the use of the HOO 2011 source documents was permitted, we believe that excluding them is more realistic.

Features	**Example**
Lexical and POS features	
Observed preposition†	*on*
Word i before ($i = 1, 2, 3$)†	{*sitting, cat, ..*}
Word i after ($i = 1, 2, 3$)†	{*the, mat, ..*}
N-grams ($N = 2, .., 5$)‡	{*sitting_X, X_the, ..* }
POS N-grams ($N = 2, 3$)	{*VBG_X, X_DT, ..* }
Head word features	
Head of prev VP†	*sitting*
POS head of prev VP	*VBG*
Head of prev NP†	*cat*
POS head of prev NP	*NN*
Head of next NP†	*mat*
POS head of next NP	*NN*
Head prev NP + head next NP†	*cat+mat*
POS head prev NP +POS head next NP	*NN+NN*
Head prev VP + head prev NP + head next NP†	*sitting+cat+mat*
POS head prev VP + POS head prev NP + POS head next NP	*VBG+NN+NN*
N-gram before + head of next NP ($N = 1, 2$)†	{*sitting+mat*}
Web N-gram count features	
Web N-gram log counts $N = 2, .., 5$	{log freq(*sitting at*), log freq(*sitting in*), .., log freq(*sitting on*), .., log freq(*sitting with*), ..}
Web dep N-gram log counts $N = 2, 3$	{log freq(*sitting-at*), log freq(*sitting-in*), .., log freq(*sitting-on*), .., log freq(*sitting-with*), .., log freq(*at-mat*), .., log freq(*on-mat*), .., log freq(*with-mat*), .., log freq(*sitting-at-mat*), ..., log freq(*sitting-on-mat*), ..}
Dependency features	
Dep parent†	*sitting*
Dep parent POS	*VBG*
Dep parent relation	*prep*
Dep child†	{*mat*}
Dep child POS	{*NN*}
Dep child relation	{*pobj*}
Dep parent+child†	*sitting+mat*
Dep parent POS+child POS†	*VBG+NN*
Dep parent+child POS†	*sitting+NN*
Dep parent POS+child†	*VBG+mat*
Dep parent+relation†	*sitting+prep*
Dep child+relation†	*mat+pobj*
Dep parent+child+relation†	*sitting+mat+prep+pobj*

Table 6.4: HOO 2012 features for replacement preposition correction. Example: "He saw a cat sitting *on the mat*." †: lexical tokens in lower case, ‡: lexical tokens in both original and lower case.

Features	**Example**
Lexical and POS features	
Word i before ($i = 1, 2, 3$)†	{*sitting, cat, ..*}
Word i after ($i = 1, 2, 3$)†	{*the, mat, ..*}
N-grams ($N = 2, .., 5$)‡	{*sitting_X, X_the, ..* }
POS N-grams ($N = 2, 3$)	{*VBG_X, X_DT, ..* }
Head word features	
Head of prev VP†	*sitting*
POS head of prev VP	*VBG*
Head of prev NP†	*cat*
POS head of prev NP	*NN*
Head of next NP†	*mat*
POS head of next NP	*NN*
Head prev NP + head next NP†	*cat+mat*
POS head prev NP + POS head next NP	*NN+NN*
Head prev VP + head prev NP + head next NP†	*sitting+cat+mat*
POS head prev VP + POS head prev NP + POS head next NP	*VBG+NN+NN*
N-gram before + head of next NP ($N = 1, 2$)†	{*sitting+mat, ..*}
Web N-gram count features	
Web N-gram log counts $N = 3, .., 5$	{log freq(*sitting on the*), log freq(*sitting the*), .. ,log freq(*sitting on the mat*), .., log freq(*sitting the mat*), ..}

Table 6.5: HOO 2012 features for missing preposition correction. Example: "He saw a cat sitting *the mat*."† : lexical tokens in lower case, ‡: lexical tokens in both original and lower case.

Features	**Example**
Web N-gram count features	
Web N-gram log counts $N = 3, .., 5$	{log freq(*went to home*), log freq(*went home*), .. ,log freq(*cat went to home*), .., log freq(*cat went home*), ..}

Table 6.6: HOO 2012 features for unwanted preposition correction. Example: "The cat went *to home*."

HOO 2012

The HOO 2012 training data consists of 1,000 documents together with gold-standard annotation. The documents are a subset of the 1,244 documents in the Cambridge Learner Corpus FCE (First Certificate in English) data set (Yannakoudakis et al., 2011). The HOO 2012 gold-standard annotation only contains edits for six determiner and preposition error types and discards all other gold edits from the original FCE data set. This can lead to "wrong" gold edits that produce ungrammatical sentences, like the following sentence

> There are a lot of possibilities ($\epsilon \rightarrow$ of) to earn some money ...

where the preposition *of* is inserted before *to earn*. The FCE data set contains another edit (*to earn* $\rightarrow$ *earning*) but this edit is not included in the HOO 2012 gold annotation. This necessarily introduces noise into the training data as a classifier trained on this data will learn that inserting *of* before *to earn* is correct. We sidestep this problem by directly using the FCE data set for training, and applying all gold edits except the six determiner and preposition error types. This gives us training data that only contains those types of grammatical errors that we are interested in. Note that this only applies to the training data. For the development and development test data, we use the HOO 2012 released data where the texts contain all types of errors and do not make use of the annotations in the FCE data set. For system development, we randomly select 100 documents from the HOO 2012 training data as the development set (HOO2012-DEV) and another 100 disjoint documents as the held-out development test set (HOO2012-DEVTEST). We train classifiers on the remaining 1,044 documents of the FCE data set (FCE(1044)), tune parameters on HOO2012-DEV, and test on HOO2012-DEVTEST. For the final HOO 2012 shared task submission, we train classifiers on all FCE documents, except those 100 documents in HOO2012-DEV which are used for parameter tuning. Finally, we fix all parameters and re-train the classifiers on the *complete* FCE corpus (FCE(1244)). This allows us to make maximum use of the FCE corpus as training data. The official HOO 2012 test data (HOO2012-TEST), which is not part of the

Data set	**# Documents**	**# Sentences**	**# Tokens**
HOO 2011			
HOO2011-TUNE	9	462	10,691
HOO2011-DEVTEST	10	477	12,115
HOO2011-TEST	18	722	18,790
ACL-ANTHOLOGY	8,618	708,129	18,020,431
CL-JOURNAL	201	22,934	611,334
HOO 2012			
FCE(1044)	1,044	22,434	339,902
FCE(1244)	1,244	28,033	423,850
HOO2012-DEV	100	2,798	42,347
HOO2012-DEVTEST	100	2,674	41,518
HOO2012-TEST	100	1,393	20,563

Table 6.7: Overview of the data sets in the HOO 2011 and HOO 2012 experiments.

FCE corpus, is completely unobserved during system development. The HOO 2011 and HOO 2012 data sets are described in more detail in Chapter 3.

Besides the FCE and HOO 2012 data sets, we use the following corpora for the HOO 2012 pipeline system. The Google Web 1T 5-gram corpus (Brants and Franz, 2006) is used for language modeling and collecting N-gram counts, the PukWaC corpus from the WaCky project (Baroni et al., 2009) is used for collecting web-scale dependency N-gram counts, and the New York Times section of the Gigaword corpus[14] and all normal-cased documents in the HOO 2012 training data are used for training the re-casing model. All data sets used in our system are publicly available. Table 6.7 gives an overview of the data sets.

6.4.2 Resources

We use the following NLP tools and resources in the pipeline system. Sentence splitting is performed with the NLTK toolkit.[15] For spelling correction, we use the free software Aspell[16]. We use the OpenNLP tools (version 1.5.2)[17] for POS tagging, YamCha (version 0.33) (Kudo and Matsumoto, 2003) for chunking, and the MaltParser (version 1.6.1) (Nivre et al., 2007) for dependency parsing. We use RandLM (Talbot and Os-

[14]LDC2009T13

[15]`http://www.nltk.org`

[16]`http://aspell.net`

[17]`http://opennlp.apache.org`

borne, 2007) for language modeling. The re-casing model for the HOO 2012 pipeline is built with the MOSES SMT system (Koehn et al., 2007). The CuVPlus English dictionary (Mitton, 1992) is used to determine the countability of nouns. The empirical risk minimization algorithm was implemented by Chang Liu (Liu and Ng, 2007). The learning algorithm was implemented by the thesis author. The source code for the CW learning algorithm is available from the NUS NLP group website[18].

6.4.3 Evaluation

Evaluation is performed by computing micro-averaged detection, recognition, and correction F_1 score between the set of system edits and the set of gold-standard edits as defined in Chapter 3 and the HOO 2011 and HOO 2012 overview papers (Dale and Kilgarriff, 2011; Dale et al., 2012). The scores are computed over the entire test collections. The official gold-standard edits are given in character offsets, while our system internally works with token offsets. Therefore, all token offsets are automatically mapped back to character offsets for evaluation.

For individual error categories, the HOO 2011 overview paper only reports the "percentage of instances in each category that were detected, recognized and corrected", but not precision or F_1 scores. Computing precision and F_1 is complicated by the fact that the HOO 2011 submission format does not require a system to "label" each proposed correction with the intended error category. For the HOO 2012 shared task, each correction had to be labeled with the appropriate error category, therefore calculating precision, recall, and F_1 score for each error category is straightforward. For the HOO 2012 shared task, we also report results for individual error categories.

For both shared tasks, evaluation on the official test sets is performed with respect to two different gold standards: the original gold standard released by the task organizers and a revised version which was created in the shared tasks in response to change requests from participating teams. For the HOO 2011 shared task, additionally each score is reported under a "with bonus" alternative, where a system receives rewards for

[18]`http://nlp.comp.nus.edu.sg/software`

Step	Detection		Recognition		Correction	
	wb	w/o b	wb	w/o b	wb	w/o b
PRE	21.52	0.00	21.52	0.00	21.52	0.0
+SPEL	22.19	0.95	21.90	0.63	21.62	0.31
+ART	26.81	10.93	25.20	9.17	24.55	8.46
+PREP	29.73	13.54	27.63	11.23	26.57	10.08

Table 6.8: **HOO 2011.** Overall F_1 scores with (wb) and without bonus (w/o b) on the HOO2011-DEVTEST data after pre-processing (PRE), spelling (SPEL), article (ART), and preposition correction (PREP).

Step	Detection		Recognition		Correction	
	wb	w/o b	wb	w/o b	wb	w/o b
PRE	13.39	0.00	13.39	0.00	13.39	0.00
+SPEL	14.57	1.36	14.20	0.97	14.02	0.78
+ART	24.94	14.73	23.30	13.05	19.80	9.34
+PREP	24.94	14.73	23.30	13.05	19.80	9.34

(a) Before revisions

Step	Detection		Recognition		Correction	
	wb	w/o b	wb	w/o b	wb	w/o b
PRE	15.53	0.00	15.53	0.00	15.53	0.00
+SPEL	16.63	0.93	16.29	0.93	16.11	0.75
+ART	27.18	15.52	25.45	13.73	22.09	10.14
+PREP	28.40	17.74	26.86	16.15	22.74	11.77

(b) After revisions

Table 6.9: **HOO 2011.** Overall F_1 scores with (wb) and without bonus (w/o b) on the HOO2011-TEST data.

missed optional edits. Scores are computed with the official scorer for the HOO 2011 and HOO 2012 shared tasks, respectively.

6.5 Results

HOO 2011

Tables 6.8 and 6.9 show the overall detection, recognition, and correction F_1 scores after each processing step on the HOO2011-DEVTEST and HOO2011-TEST set, respectively. Each processing step builds on the output of the previous step. The single biggest improvement in the score comes from the article correction step. The gap between the scores with and without bonus shows the large number of optional corrections in the HOO 2011 data. Our pipeline system achieved the second highest F_1 correction score on the official HOO 2011 test set (Dale and Kilgarriff, 2011).

Step	Recognition			Correction		
	P	R	F_1	P	R	F_1
Det	62.26	12.68	21.06	54.09	11.01	18.30
+ RT	64.34	22.41	33.24	57.35	19.97	29.63
+ MT/UT	60.75	28.94	**39.20**	54.84	26.12	**35.39**

Table 6.10: **HOO 2012.** Overall precision, recall, and F_1 score on the HOO2012-DEVTEST data after article correction (Det), replacement preposition correction (RT), and missing and unwanted preposition correction (MT/UT).

HOO 2012

Tables 6.10 and 6.12 show the overall precision, recall and F_1 score of the system after each processing step on the held-out HOO2012-DEVTEST set and the official HOO 2012 test set, respectively. All numbers are shown in percentages. We note that each processing step improves the overall performance. Tables 6.11 and 6.13 show individual precision, recall, and F_1 score for each of the six error types, and for articles (Det: aggregate of RD, MD, UD) and prepositions (Prep: aggregate of RT, MT, UT) on the held-out HOO2012-DEVTEST set and the official test set HOO2012-TEST, respectively. The final F_1 correction score achieved by our pipeline system on the official HOO 2012 test set is 28.70% before revision and 37.83% after revision, which are the highest scores achieved by any participating team in the shared task (Dale et al., 2012).

Type	Recognition			Correction		
	P	R	F_1	P	R	F_1
RD	30.00	5.66	9.52	30.00	5.66	9.52
MD	69.67	41.67	52.15	59.02	35.29	44.17
UD	40.74	11.00	17.32	40.74	11.00	17.32
Det	62.26	27.73	38.37	54.09	24.09	33.33
RT	69.09	33.63	45.24	63.64	30.97	41.67
MT	53.25	35.34	42.49	49.35	32.76	39.38
UT	38.46	12.20	18.52	38.46	12.20	18.52
Prep	59.62	29.95	39.87	55.40	27.83	37.05

Table 6.11: **HOO 2012.** Individual scores for each error type on the HOO2012-DEVTEST data.

6.6 Discussion

The main differences between the systems for the HOO 2011 shared task and the HOO 2012 shared task are the use of the CW learning algorithm, the use of web-scale N-gram

Step	Recognition			Correction		
	P	R	F_1	P	R	F_1
Det	57.76	14.79	23.55	48.28	12.36	19.68
+ RT	58.93	21.85	31.88	47.02	17.44	25.44
+ MT/UT	55.98	25.83	**35.35**	45.45	20.97	**28.70**

(a) Before revisions

Step	Recognition			Correction		
	P	R	F_1	P	R	F_1
Det	68.10	16.70	26.83	62.93	15.43	24.79
+ RT	71.43	25.37	37.44	63.10	22.41	33.07
+ MT/UT	69.38	30.66	**42.52**	61.72	27.27	**37.83**

(b) After revisions

Table 6.12: **HOO 2012.** Overall precision, recall, and F_1 score on the HOO2012-TEST data after article correction (Det), replacement preposition correction (RT), and missing and unwanted preposition correction (MT/UT).

Type	Recognition			Correction		
	P	R	F_1	P	R	F_1
RD	33.33	2.56	4.76	33.33	2.56	4.76
MD	62.24	48.80	54.71	51.02	40.00	44.84
UD	33.33	9.43	14.71	33.33	9.43	14.71
Det	57.76	30.88	40.24	48.28	25.81	33.63
RT	61.54	23.53	34.04	44.23	16.91	24.47
MT	46.15	21.05	28.92	38.46	17.54	24.10
UT	40.00	13.95	20.69	40.00	13.95	20.69
Prep	53.76	21.19	30.40	41.94	16.53	23.71

(a) Before revisions

Type	Recognition			Correction		
	P	R	F_1	P	R	F_1
RD	100.00	8.33	15.38	66.67	5.56	10.26
MD	70.41	52.67	60.26	65.31	48.85	55.90
UD	46.67	11.29	18.18	46.67	11.29	18.18
Det	68.10	34.50	45.80	62.93	31.88	42.32
RT	78.85	27.52	40.80	63.46	22.15	32.84
MT	61.54	28.57	39.02	53.85	25.00	34.15
UT	60.00	23.08	33.33	60.00	23.08	33.33
Prep	70.97	27.05	39.17	60.22	22.95	33.23

(b) After revisions

Table 6.13: **HOO 2012.** Individual scores for each error type on the HOO2012-TEST data.

count features, and the use of the observed article or preposition as a feature. The CW learning algorithm performed slightly better than the empirical risk minimization batch learning algorithm while being significantly faster during training. Adding the web-scale N-gram count features showed significant improvements in initial experiments. Using the observed article or preposition feature allows the classifier to learn a bias against unnecessary corrections. We believe that the good precision scores are a result of using this feature.

In our experiments with the HOO 2012 data, we tried adding additional training data from other text corpora: the NUCLE corpus from Chapter 3 and the Gigaword corpus. Unfortunately, we did not see any consistent improvements over simply using the FCE corpus. The general rule of thumb that "more data is better data" did not seem to hold true in this case. After the evaluation had completed, we also tried training on additional training data and tested the resulting system on the official test set but did not see improvements either. We believe that no improvements were obtained due to the similarity between the training and test data, since all of them are student essays written in response to question prompts from the Cambridge FCE exam.

6.7 Conclusion

In this chapter, we have presented a pipeline architecture for grammatical error correction that combines multiple correction steps into an end-to-end correction system. This architecture was used in the NUS systems participating in the HOO 2011 and HOO 2012 shared tasks. Our systems achieves the second highest correction F_1 score on the HOO 2011 official test and the highest correction F_1 score on the HOO 2012 official test set among all participating teams.

Chapter 7

A Beam-Search Decoder for Grammatical Error Correction

7.1 Introduction

The dominant paradigm that underlies most existing grammar correction systems today is supervised classification. For each error category, a multi-class classifier is trained to predict a word from a *confusion set* of possible correction choices, given some feature representation of the surrounding sentence context. During testing, each classifier predicts the most likely correction for each test instance. If the prediction differs from the observed word used by the writer and the classifier is sufficiently confident in its prediction, the observed word is replaced by the prediction. Multiple correction steps are combined in an ad-hoc manner. For example, in the pipeline architecture presented in the last chapter, after correcting all errors of a particular category, the corrected output is fed into the next correction step. The output of the last correction step is the final output of the system. This approach has been followed by the vast majority of error correction systems that have been proposed to date.

Although considerable progress has been made, the classifier-based approach suffers from some serious shortcomings. Each classifier corrects a single word for a specific error category individually. This ignores dependencies between the words in a sentence. Also, by conditioning on the surrounding context, the classifier implicitly assumes that the surrounding context is free of grammatical errors, which is often not the case. Finally, the classifier has to commit to a single one-best prediction and is not able to change its decision later or explore multiple corrections. Instead of correcting each word individually, it would be preferable to perform global inference over corrections of whole sentences which can contain multiple and interacting errors.

An alternative paradigm is to view error correction as a statistical machine translation (SMT) problem from "bad" to "good" English. The system for correction of lexical choice errors in Chapter 5 showed one way how a standard SMT system could be used to correct grammatical errors. However, the system only focused on a single error category, lexical choice errors, and assumed that the identification of lexical choice errors had already been performed. While the SMT approach can in principle correct whole sentences and multiple error categories, a standard SMT system cannot easily incorporate models for specific grammatical errors. It also suffers from the paucity of error-annotated training data for grammar correction. As a result, applying a standard SMT system to error correction does not produce good results, as we show in this chapter.

In this chapter, we present a novel beam-search decoder for grammatical error correction that combines the advantages of the classification approach and the SMT approach. Starting from the original input sentence, the decoder performs an iterative search over possible sentence-level hypotheses to find the best sentence-level correction. In each iteration, a set of *proposers* generates new hypotheses by making incremental changes to the hypotheses found so far. A set of *experts* scores the new hypotheses on criteria of grammatical correctness. These experts include discriminative classifiers for specific error categories, such as articles and prepositions. The decoder model calculates the overall hypothesis score for each hypothesis as a linear combination of the expert scores. The weights of the decoder model are discriminatively trained on a development set of error-annotated sentences. The highest scoring hypotheses are kept in the search beam for the next iteration. This search procedure continues until the beam is empty or the maximum number of iterations has been reached. The highest scoring hypothesis is returned as the sentence-level correction. We evaluate our proposed decoder in the context of the HOO 2011 and HOO 2012 shared task on grammatical error correction (Dale and Kilgarriff, 2011; Dale et al., 2012). Our decoder improves upon a baseline system that uses the pipeline architecture presented in the last chapter and establishes new state-of-the-art result on both data sets. Initial results of this work on the HOO 2011 data were published in (Dahlmeier and Ng, 2012a).

The remainder of this chapter is organized as follows. Section 7.2 describes the proposed beam-search decoder. Sections 7.3 and 7.4 describe the experimental setup and results, respectively. Section 7.5 provides further discussion. Section 7.6 concludes the chapter.

7.2 Decoder

In this section, we describe the proposed beam-search decoder and its components.

The task of the decoder is to find the best hypothesis (i.e., the best corrected sentence) for a given input sentence. To accomplish this, the decoder needs to be able to perform two tasks: generating new hypotheses from current ones, and discriminating good hypotheses from bad ones. This is achieved by two groups of modules which we call *proposers* and *experts*, respectively. Proposers take a hypothesis and generate a set of new hypotheses, where each new hypothesis is the result of making an incremental change to the current hypothesis. Experts score hypotheses on particular aspects of grammaticality. This can be a general language model score or the output of classifiers for particular error categories, for example for article and preposition usage. The overall score for a hypothesis is a linear combination of the expert scores. Note that in our decoder, each hypothesis corresponds to a complete sentence. This makes it easy to apply syntactic processing, like part-of-speech (POS) tagging, chunking, and dependency parsing, which provides necessary features for the expert models. The highest scoring hypotheses are kept in the search beam for the next iteration. The search ends when the beam is empty or the maximum number of iterations has been reached. The highest scoring hypothesis found during the search is returned as the sentence-level correction. The modular design of the decoder makes it easy to extend the model to new error categories by adding specific proposers and experts without having to change the decoding algorithm.

7.2.1 Proposers

The proposers generate new hypotheses, given a hypothesis. Because the number of possible hypotheses grows exponentially with the sentence length, enumerating all possible hypotheses is infeasible. Instead, each proposer only makes a small incremental change to the hypothesis in each iteration. A change corresponds to a correction of a single word or phrase. We experiment with the following proposers in this chapter. Additional proposers for other error categories can easily be added to the decoder.

- **Spelling** Generate a set of new hypotheses by replacing a misspelled word with each correction proposed by a spellchecker.

- **Articles** For each noun phrase (NP), generate two new hypotheses by changing the observed article. Possible article choices are *a/an*, *the*, and the null article ϵ.

- **Preposition replacement** For each prepositional phrase (PP), generate a set of new hypotheses by changing the observed preposition. For each preposition, we define a confusion set of possible corrections. Prepositions inserted by the decoder (see below) are not replaced.

- **Preposition insertion** For each NP that is not preceded by a preposition, generate a set of new hypotheses by inserting a preposition before the NP. Skip NPs whose preceding preposition has previously been deleted.

- **Preposition deletion** For each PP, generate a new hypothesis by deleting the preposition.

- **Punctuation insertion** Insert commas, periods, and hyphens based on a set of simple rules. Skip prepositions that have been inserted by the decoder.
- **Noun number** For each noun, change its number from singular to plural or vice versa.

7.2.2 Experts

The experts score hypotheses on particular aspects of grammaticality to help the decoder to discriminate grammatical hypotheses from ungrammatical ones. We employ two types of expert models. The first type of expert model is a standard N-gram language model. The language model expert is not specialized for any particular type of error. The second type of experts is based on linear classifiers and is specialized for particular error categories. We use the following classifier experts in our work. The features for the classifier expert models include features from N-grams, POS tags, chunks, web-scale N-gram counts, and dependency parse trees. Additional experts can easily be added to the decoder.

- **Article expert** Predict the correct article for a noun phrase. Pronouns and proper nouns are excluded.
- **Preposition replacement expert** Predict the correct preposition for a prepositional phrase. This expert does not score prepositions that were inserted by the decoder.
- **Preposition insertion expert** Predict whether a particular preposition should be inserted in front of a noun phrase or not. Noun phrases that already are part of a prepositional phrase are skipped. We use a separate preposition insertion expert for each preposition.
- **Preposition deletion expert** Predict whether a particular preposition in front of a noun phrase should be deleted or not. If the original preposition used by the writer has been replaced by the preposition replacement proposer, the original

preposition is considered as not deleted. We use a separate preposition deletion expert for each preposition.

- **Noun number expert** Predict whether a noun should be in the singular or plural form.

The outputs of the experts are used as hypothesis features in the decoder, as described in the next section.

7.2.3 Hypothesis Features

Each hypothesis is associated with a vector of real-valued features which are indicators of grammaticality and are computed from the output of the expert models. We call these features *hypothesis features* to distinguish them from the features of the expert classifiers. The simplest hypothesis feature is the log probability of the hypothesis under the N-gram language model expert. To avoid a bias towards shorter hypotheses, we normalize the probability by the length of the hypothesis:

$$score_{lm} = \frac{1}{|\mathbf{h}|} \log Pr(\mathbf{h}), \tag{7.1}$$

where $\mathbf{h}$ is a hypothesis sentence and $|\mathbf{h}|$ is the hypothesis length in tokens.

For the classifier-based experts, we define two types of features. The first is the *average score* of the hypothesis under an expert model:

$$score_{avg} = \frac{1}{n} \sum_{i=1}^{n} \left(\mathbf{u}^T f(\mathbf{x_i^h}, y_i^{\mathbf{h}}) \right), \tag{7.2}$$

where $\mathbf{u}$ is the expert classifier weight vector, $\mathbf{x_i^h}$ and $y_i^{\mathbf{h}}$ are the feature vector and the hypothesis class, respectively, for the i-th instance extracted from the hypothesis $\mathbf{h}$ (e.g., the feature vector and the article for the i-th NP in the hypothesis for the article expert), and f is a feature map that computes the expert classifier features. The average score reflects how much the expert model "likes" the hypothesis. The second expert score, which we call *delta score*, is the maximum difference between the highest scoring class and the hypothesis class in any instance from the hypothesis:

$$score_{delta} = \max_{i,y} \left(\mathbf{u}^T f(\mathbf{x_i^h}, y) - \mathbf{u}^T f(\mathbf{x_i^h}, y_i^{\mathbf{h}}) \right). \tag{7.3}$$

Generally speaking, the delta score measures how much the model "disagrees" with the hypothesis.

Finally, each hypothesis has a number of *correction count features* that keep track of how many corrections have been made to the hypothesis so far. For example, there is a feature that counts how often the article correction $\epsilon \rightarrow the$ has been applied. We also add aggregated correction count features for each error category, e.g., how many article corrections have been applied in total. The correction count features allow the decoder to learn a bias against over-correcting sentences and to learn which types of corrections are more likely and which are less likely.

7.2.4 Decoder Model

The hypothesis features described in the previous subsection are combined to compute the score of a hypothesis according to the following linear model:

$$s = \mathbf{w}^T f_E(\mathbf{h}), \tag{7.4}$$

where $\mathbf{w}$ is the decoder model weight vector and f_E is a feature map that computes the hypothesis features described above, given a set of experts E.

To illustrate the process of scoring a hypothesis, consider the following real example from our HOO 2012 experiments described in Section 7.3. The original input sentence *I hope and I wish that the future will bring us a good moments of life where everybody will have home and a warm, friendly family relationships.* contains three article errors and one preposition replacement error: deleting the indefinite article *a* before *good moment* and *warm, friendly family relationships*, inserting *a* before *home*, and changing the preposition *of* to *in*. The best hypothesis sentence found by the decoder was *I hope and I wish that the future will bring us good moments in life where everybody will have* *home and warm, friendly family relationships.* which successfully corrects the preposition error and two of the three article errors. The associated hypothesis features are computed as follows. The language expert score is the normalized log probability of the hypothesis sentence. The article expert score is the sum of the article classifier scores for the article *the* before the NP *future* and the null article before the NPs *good moments*, *life*, *everybody*, *home*, and *warm, friendly family relationships* divided by the number of NPs. The pronouns *I* and *us* are not considered. We note that the article expert classifier's observed article feature is fixed to the article used by the writer in the original sentence, even if the article proposer changes the article in the current hypothesis. The preposition replacement expert average score is the preposition replacement classifier score for the preposition *in* before *life*. As there is only one prepositional phrase, the average consists of a single term in this case. The observed preposition feature for the preposition expert classifier is the preposition used by the writer in the original sentence, even if the preposition replacement proposer has changed the preposition. For the preposition insertion experts, the average score is the average of the scores given by the preposition insertion classifier for the "null preposition" before the NPs *I*, *I*, *future*, *us*, *good moments*, *everybody*, *home*, and *warm, friendly family relationships*. The NP *life* is not considered for preposition insertion as it already has a preposition. For the preposition deletion experts, there is only the prepositional phrase *of life* that was changed to *in life*. As the preposition *of* was not deleted but replaced with another preposition, this prepositional phrase is an example for the *of* preposition deletion expert with the class label "no deletion" and the preposition deletion expert average score is the preposition deletion classifier score for not deleting this preposition. The expert delta scores are zero for all experts except for the preposition insertion expert for the preposition *to*. The correction count features keep track of the two article corrections and the one preposition replacement correction. All features are normalized to a unit interval to avoid having features on a larger scale dominate features on a smaller scale.

We linearly scale all hypothesis features to a unit interval $[0, 1]$

$$score = \frac{score_{orig} - score_{min}}{score_{max} - score_{min}}. \tag{7.5}$$

The minimum and maximum values for each feature are estimated from the development data. The normalized features are combined with the decoder model weight vector $\mathbf{w}^T$ to compute the over hypothesis score. The hypothesis features are shown in Table 7.1. Note that due to the scaling, features that have zero values before the scaling can be non-zero after scaling. The absolute value of the features are not important in ranking the hypothesis, only the relative difference matters.

PRO Tuning

The weight vector $\mathbf{w}$ is tuned on a development set of error-annotated sentences using the PRO ranking optimization algorithm (Hopkins and May, 2011).[19] PRO performs decoder parameter tuning through a pair-wise ranking approach. The algorithm starts by sampling hypothesis pairs $(\mathbf{h_i}, \mathbf{h_j})$ from the N-best list of the decoder. The metric score $g(\mathbf{h})$ for each hypothesis induces a ranking of the two hypotheses in each pair. The goal of the PRO algorithm is to find a weight vector $\mathbf{w}$ such that the hypothesis scores rank the hypothesis pair in the same order as the metric scores:

$$g(\mathbf{h_i}) > g(\mathbf{h_j}) \Leftrightarrow \mathbf{w}^T f_E(\mathbf{h_i}) > \mathbf{w}^T f_E(\mathbf{h_j}). \tag{7.6}$$

The task of finding a weight vector that correctly ranks hypotheses can then be reduced to a simple binary classification task.

$$\begin{aligned} g(\mathbf{h_i}) > g(\mathbf{h_j}) &\Leftrightarrow \mathbf{w}^T f_E(\mathbf{h_i}) > \mathbf{w}^T f_E(\mathbf{h_j}) \\ &\Leftrightarrow \mathbf{w}^T f_E(\mathbf{h_i}) - \mathbf{w}^T f_E(\mathbf{h_j}) > 0 \\ &\Leftrightarrow \mathbf{w}^T (f_E(\mathbf{h_i}) - f_E(\mathbf{h_j})) > 0 \end{aligned} \tag{7.7}$$

[19]We also experimented with the MERT algorithm (Och, 2003) but found that PRO achieved better results.

Original	I hope and I wish that the future will bring us a good moments of life where everybody will have home and a warm , friendly family relationships .
Gold	I hope and I wish that the future will bring us (*a* $\rightarrow \epsilon$) good moments (*of* $\rightarrow$ *in*) life where everybody will have ($\epsilon \rightarrow$ *a*) home and *a* $\rightarrow \epsilon$) warm , friendly family relationships .
Hypothesis	I hope and I wish that the future will bring us (*a* $\rightarrow \epsilon$) good moments (*of* $\rightarrow$ *in*) life where everybody will have home and (*a* $\rightarrow \epsilon$) warm , friendly family relationships .

Hypothesis features	**Un-scaled**	**Scaled**
Language model expert $score_{lm}$	-5.0294	0.8575
Article expert $score_{avg}$	0.1019	0.3474
Article expert $score_{delta}$	0.0	0.0
Article corrections	2	0.6667
Article corrections $a \rightarrow \epsilon$	2	0.6667
Preposition replacement expert $score_{avg}$	0.2800	0.6264
Preposition replacement expert $score_{avg}$	0.0	0.0
Preposition replacement corrections	1	0.3333
Preposition replacement corrections $of \rightarrow in$	1	0.3333
Preposition insertion expert (to) $score_{avg}$	0.2319	0.6585
Preposition insertion expert (to) $score_{delta}$	0.0109	0.0116
Preposition insertion expert (for) $score_{avg}$	0.1897	0.8433
Preposition insertion expert (at) $score_{avg}$	0.2423	0.8264
Preposition insertion expert (on) $score_{avg}$	0.2665	0.7764
Preposition insertion expert (in) $score_{avg}$	0.2261	0.6965
Preposition insertion expert (of) $score_{avg}$	0.3108	0.7472
Preposition insertion expert (about) $score_{avg}$	0.3009	0.8559
Preposition deletion expert (to) $score_{avg}$	0.0	0.3355
Preposition deletion expert (for) $score_{avg}$	0.0	0.1565
Preposition deletion expert (at) $score_{avg}$	0.0	0.3239
Preposition deletion expert (on) $score_{avg}$	0.0	0.4426
Preposition deletion expert (in) $score_{avg}$	0.0	0.2439
Preposition deletion expert (of) $score_{avg}$	0.0838	0.6045
Preposition deletion expert (of) $score_{avg}$	0.0	0.1255
Hypothesis score	12.20408	

Table 7.1: Examples of a source sentence, generated hypothesis, and hypothesis features. Most zero-valued scaled hypothesis features are omitted because of space constraint.

For each hypothesis pair, a labeled training example is created from the difference feature vector $f_E(\mathbf{h_i}) - f_E(\mathbf{h_j})$. An example is labeled as a positive example if $g(\mathbf{h_i}) > g(\mathbf{h_j})$ and as a negative example otherwise. To keep the training set balanced, a second labeled example is created from the swapped hypothesis pair. The training set is used as input to a standard linear classifier learning algorithm that returns a weight vector $\mathbf{w}$ optimized on the above hypothesis pairs. The weight vector can theoretically directly be used in the decoder for the next iteration. However, to explicitly tie the weight vector in the t-th iterations to the weights of the previous $t-1$-th iteration and to make the tuning process more stable, the new weight vector $\mathbf{w}$ is interpolated with the weight vector $\mathbf{w}_{t-1}$ from the previous iteration to form the new weight vector $\mathbf{w}_t$.

$$\mathbf{w}_t = \lambda \cdot \mathbf{w} + (1 - \lambda) \cdot \mathbf{w}_{t-1} \tag{7.8}$$

The tuning process continues until the maximum number of iterations is reached or some early stopping criterion is met.

In this work, we use PRO to optimize the F_1 correction score, which is defined in Chapter 3. PRO requires a sentence-level score for each hypothesis. As F_1 score is not decomposable, we optimize sentence-level F_1 score which serves as an approximation of the corpus-level F_1 score. Similarly, Hopkins and May optimize a sentence-level BLEU approximation (Lin and Och, 2004) instead of the corpus-level BLEU score (Papineni et al., 2002). We observed that optimizing sentence-level F_1 score worked well in practice in our experiments.

7.2.5 Decoder Search

Given a set of proposers, experts, and a tuned decoder model, the decoder can be used to correct new unseen sentences. This is done by performing a search over possible hypothesis candidates. The decoder starts with the input sentence as the initial hypothesis, i.e., assuming that all words are correct. It then performs a beam search over the space of possible hypotheses to find the best hypothesis correction $\hat{\mathbf{h}}$ for an input sentence $\mathbf{e}$.

The search proceeds in iterations until the beam is empty or the maximum number of iterations has been reached. In each iteration, the decoder takes each hypothesis in the beam and generates new hypothesis candidates using all the available proposers. The hypotheses are evaluated by the expert models. For each newly generated hypothesis, the decoder performs POS tagging, chunking, and dependency parsing, followed by feature extraction, and computation of the average and delta hypothesis features for each classifier expert model. Finally, each hypothesis is scored using the decoder model. As the search space grows exponentially, it is infeasible to perform exhaustive search. Therefore, we prune the search space by only accepting the most promising hypotheses to the pool of hypotheses for future consideration. If a hypothesis has a higher score than the best hypothesis found in previous iterations, it is definitely added to the pool. Otherwise, we use a simulated annealing strategy where hypotheses with a lower score can still be accepted with a certain probability which depends on the difference between the hypothesis score and the score of the best hypothesis, and the "temperature" of the system. We lower the temperature after each iteration according to an exponential cooling schedule. Hypotheses that have been explored before are not considered again to avoid cycles in the search. From all hypotheses in the pool, we select the top k hypotheses and add them to the beam for the next search iteration. The decoding algorithm is shown in Algorithm 1. The decoder can be considered an *anytime algorithm* (Russell and Norvig, 2010), as it has a current best hypothesis correction available at any point of the search, while gradually improving the result by searching for better hypotheses. An example of a search tree produced by the decoder is shown in Figure 7.1.

During the search, the expert model scores are re-computed for every hypothesis because the features that are input to the expert models change when words in the hypothesis are changed, for example a preposition can be part of the N-gram features that feed into the article expert model. In general, we expect that "better" hypotheses should receive higher scores from *all* hypothesis features, e.g., the score computed by the article expert should increase indirectly if the preposition that is part of the feature input is correct. In certain cases, however, there can be "cross talk" between the expert models

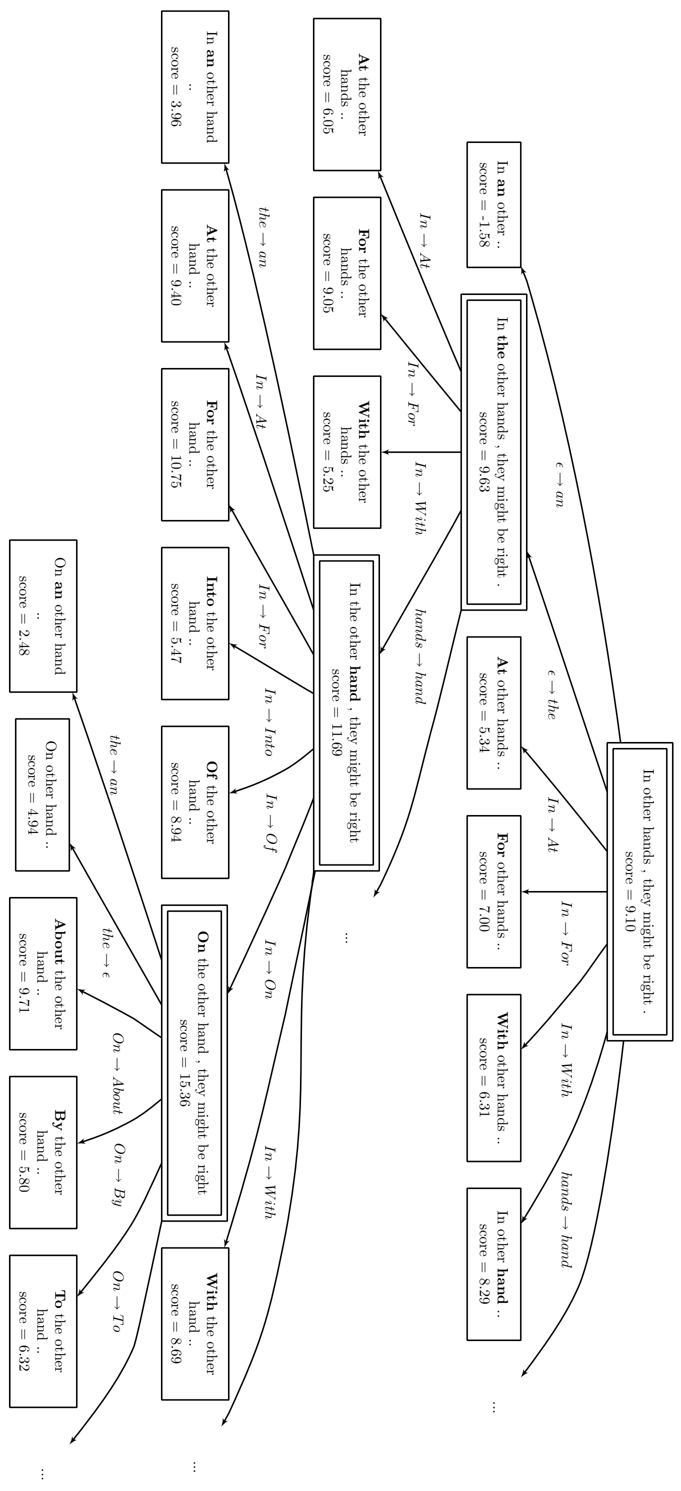

In other hands , they might be right . score = 9.10
$\epsilon \to an$
$\epsilon \to the$
$In \to At$
$In \to For$
$In \to With$
$hands \to hand$
In **an** other .. score = -1.58
In **the** other hands , they might be right . score = 9.63
At other hands .. score = 5.34
For other hands .. score = 7.00
With other hands .. score = 6.31
In other **hand** .. score = 8.29
At the other hands .. score = 6.05
For the other hands .. score = 9.05
With the other hands .. score = 5.25
In the other **hand** , they might be right score = 11.69
$the \to an$
$In \to Into$
$In \to Of$
$In \to On$
In **an** other hand .. score = 3.96
At the other hand .. score = 9.40
For the other hand .. score = 10.75
Into the other hand .. score = 5.47
Of the other hand .. score = 8.94
On the other hand , they might be right score = 15.36
With the other hand .. score = 8.69
$the \to \epsilon$
$On \to About$
$On \to By$
$On \to To$
On **an** other hand .. score = 2.48
On other hand .. score = 4.94
About the other hand .. score = 9.71
By the other hand .. score = 5.80
To the other hand .. score = 6.32

Figure 7.1: Example of a search tree produced by the beam-search decoder. Some hypotheses are omitted due to space constraints.

where a change that is bad according to one expert is good according to another expert. In our experiments, this happened for insertion and deletion of prepositions which negatively influenced article corrections. For example, in the sentence, *I am interested in tennis*, the article expert gives a higher score for keeping the null article before the NP *tennis* than for inserting the article *the*, which is correct. However, the preposition deletion expert gives a higher score for not deleting the preposition *in* when the article *the* is inserted before *tennis* compared to the score for the original sentence, presumably because of the frequent bigram *in the*. Although the article expert prefers the sentence *I am interested in tennis*, the preposition deletion model overpowers the article expert to produce the final output *I am interested in the tennis*, which is wrong. Ideally, we want the preposition insertion and deletion experts to only score the presence and absence of prepositions and not the choice of articles. To overcome this problem, we change the way the preposition insertion and deletion hypothesis scores are computed. The scores for these two experts are cached during the decoding process and are only updated if a preposition was inserted or deleted. Otherwise, the previously cached scores are used. Another challenge with preposition insertion and deletion is that the number of instances for the preposition replacement expert changes due to insertion and deletion. To allow a fair comparison of preposition replacement expert scores between hypotheses, prepositions inserted by the decoder are ignored by the preposition replacement expert, and the preposition replacement classifier scores for deleted prepositions are still included when computing the preposition replacement expert scores. In other words, the number of prepositions considered for the preposition replacement expert scores is kept constant.

The decoding algorithm shares some similarities with the beam-search algorithm frequently used in SMT. There are, however, some differences between SMT decoding and the grammar correction decoding algorithm presented here that are worth pointing out. In SMT decoding, every input word needs to be translated exactly once. In contrast, in grammar correction decoding the majority of the words typically do *not* need any correction (in the HOO 2011 data, for example, there are on average 6 errors per 100 words). On the other hand, some words might require *multiple* corrections, for example spelling correction followed by noun number correction. Errors can also be inter-dependent, where correcting one word makes it necessary to change another word, for example to preserve agreement. Our decoding algorithm has the option to correct some words multiple times, while leaving other words unchanged. Another difference concerns the hypothesis features. In phrase-based SMT, the features, e.g., phrase translation probability, are required to factor over the phrase segmentation of the sentence, that means the features can be computed independently of the context of the phrase (except for the language model). That makes it possible to pre-compute these features for each phrase and store them in a phrase table. In our case, we want to condition on arbitrary features from the continuously changing hypothesis sentence without making strong independence assumptions. While this makes it necessary to re-compute features for each hypothesis, it allows us to include more expressive features like the expert classifier scores. The decoding algorithm presented here is also different from SMT lattice decoding (Dyer et al., 2008), which models different alternatives in the input to the decoder, e.g., different word segmentation standards, and requires us to make the same strong independence assumptions as phrase-based SMT itself.

A similar argument applies to confusion networks and lattice-based methods that were developed in speech recognition (Jelinek, 1998). A lattice is a directed acyclic graph that serves as a compact representation of a potentially exponential number of hypotheses. A confusion network is a special case of a lattice where all paths through the lattice have the same length. Lattices or confusion networks can be used for error correction by taking the original sentence as the backbone of the lattice and adding possible corrections of words or phrases as additional arcs in the lattice (Lee and Seneff, 2006). Each arc can be associated with one or more local feature values, e.g., the probability of substituting a correction for the original word. The lattice is then decoded with a language model to find the best path through the lattice. The words on the path edges are the corrected sentence. The lattice is similar to the search graph that is explored by a phrase-based SMT system during monotone decoding. Note that the lattice decoding described here is different from SMT lattice decoding (Dyer et al., 2008) where the words on the lattice edges do not form the output of the decoding but the *input* to a subsequent translation step.

To illustrate the difference between lattice decoding and the beam-search decoder presented here, consider the example shown in Figure 7.2. The input sentence *All boyy play football .* contains a spelling mistake and a noun number error on the word *boyy* which should be *boys* to be in agreement with the quantifier *all* and the verb *play*. The lattice adds two possible correction arcs to the word: one for forming a plural noun and one for correcting the spelling mistake. Although the lattice can correct the spelling mistake, it cannot correct the same word again to fix the resulting noun number agreement error. To correct the mistake, the lattice would have to add an arc *boys*. Generating this arc would require to enumerate all combinations of individual corrections, e.g., all combinations of spelling corrections and noun number corrections. This would result in a very large number of edges. In contrast, the decoder can propose a spelling correction from *boyy* to *boy* and subsequently propose a noun number correction from *boy* to *boys*.

Even if words which require multiple corrections may not be very frequent, the ability to correct these mistakes is a principal advantage of the presented beam-search decoding strategy. Another advantage of the decoder method is that the whole sentence context is available and existing classifiers can easily be used to compute expert scores for particular error categories. For example, it is easy to apply a classifier to score the noun number form of *boy* using arbitrary features from the sentence context. For the lattice, it is not possible to apply the same classifiers to compute edge features as the context of an edge is not known beforehand. Even during the lattice decoding process, only the left context is available. The right context, which contains important information, like the head word of a following constituent, cannot be used to compute features in lattice decoding.

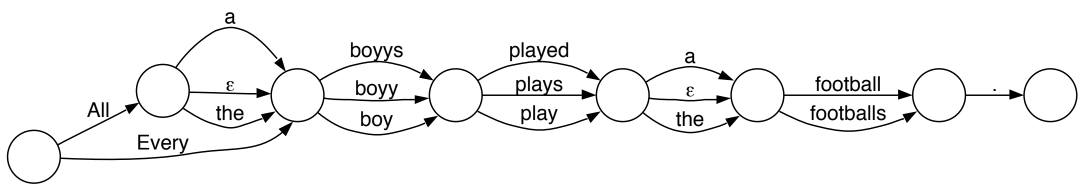


Figure 7.2: Example of a lattice for error correction. Unlike the decoder method, the lattice cannot correct the misspelled word *boyys* to *boy* and subsequently correct the resulting noun number agreement error.

7.3 Experiments

We evaluate the decoder in the context of the HOO 2011 and HOO 2012 shared tasks on grammatical error correction. We compare our proposed method with a pipeline of rule-based and classifier-based correction steps as it was described in the last chapter. For the HOO 2011 shared task, we additionally include a baseline based on phrase-based SMT. The details of the experimental setup differ slightly from the last chapter. For the HOO 2011 shared task, we discard the HOO2011-DEVTEST set and directly evaluate on the official HOO 2011 test set. For the HOO 2012 shared task, we only evaluate on the HOO2012-DEVTEST set. The official HOO 2012 test data had to be deleted right after the shared task due to copyright reasons. Thus, an evaluation on the official HOO 2012 test set was unfortunately impossible.

7.3.1 Data Sets

HOO 2011

For the experiments on the HOO 2011 data set, we split the HOO 2011 development data into an equal sized training (HOO2011-TRAIN) and tuning (HOO2011-TUNE) set. The split is performed on the sentence level. The first half of each document in the development data is assigned to the training data, and the second half of each document is assigned to the tuning data. Note that this differs from the previous chapter, where the HOO 2011 development data was split on the file level. As a result the sentence and token counts for the splits in Table 6.7 and Table 7.2 differ slightly. Splitting the data at

Algorithm 1 The beam-search decoding algorithm. $\mathbf{e}$: original sentence, $\mathbf{w}$: decoder weight vector, P: set of proposers, E: set of experts, k: beam width, M: maximum number of iterations, T, c: initial temperature and cooling schedule for simulated annealing ($0 < c < 1$).

```
procedure decode(e, w, P, E, k, M)
 1: $beam \leftarrow \{\mathbf{e}\}$
 2: $previous \leftarrow \{\mathbf{e}\}$
 3: $\mathbf{h}_{best} \leftarrow \mathbf{e}$
 4: $s_{best} \leftarrow \mathbf{w}^T f_E(\mathbf{h}_{best})$
 5: $i \leftarrow 0$
 6: while $beam \neq \emptyset \wedge i < M$ do
 7:     $pool \leftarrow \{\}$
 8:     for all $\mathbf{h} \in beam$ do
 9:         for all $p \in P$ do
10:             for all $\mathbf{h}' \in p.propose(h)$ do
11:                 if $\mathbf{h}' \in previous$ then
12:                     $continue$
13:                 $previous \leftarrow previous \cup \{\mathbf{h}'\}$
14:                 $s_{\mathbf{h}'} \leftarrow \mathbf{w}^T f_E(\mathbf{h}')$
15:                 if $accept(s_{\mathbf{h}'}, s_{best}, T)$ then
16:                     $pool \leftarrow pool \cup \{(\mathbf{h}', s_{\mathbf{h}'})\}$
17:     $beam \leftarrow \emptyset$
18:     for all $(\mathbf{h}, s_{\mathbf{h}}) \in nbest(pool, k)$ do
19:         $beam \leftarrow beam \cup \{\mathbf{h}\}$
20:         if $s_{\mathbf{h}} > s_{best}$ then
21:             $\mathbf{h}_{\mathbf{best}} \leftarrow \mathbf{h}$
22:             $s_{best} \leftarrow s_{\mathbf{h}}$
23:     $T \leftarrow T \times c$
24:     $i \leftarrow i + 1$
25: return $\mathbf{h}_{best}$

procedure accept($s_{\mathbf{h}}$, $s_{best}$, $T$)
 1: $\delta \leftarrow s_{\mathbf{h}} - s_{best}$
 2: if $\delta > 0$ then
 3:     return $true$
 4: if $exp(\frac{\delta}{T}) > random()$ then
 5:     return $true$ else return $false$
```

the file level is generally the preferred option, but splitting each document and assigning one part to the training and one part to the tuning set resembles how the training and test data for the HOO 2011 task were created.

As in the last chapter, the official HOO 2011 test data (HOO2011-TEST) is used for evaluation with both the original and the final official gold-standard annotations. The final official gold standard includes changes made in response to objections made by participants in the shared task after the test data was released. As a result, the final offi-

Data Set	**Sentences**	**Tokens**
HOO 2011		
HOO2011-TRAIN	467	11,373
HOO2011-TUNE	472	11,435
HOO2011-TEST	722	18,790
ACL-ANTHOLOGY	943,965	22,465,690
HOO 2012		
FCE(1044)	22,434	339,902
HOO2012-DEV	2,798	42,347
HOO2012-DEVTEST	2,674	41,518

Table 7.2: Overview of the HOO 2011 and HOO 2012 data sets.

cial gold-standard annotations could be biased in favor of specific systems participating in the shared task.

We use the ACL Anthology as training data for the expert models. We crawl all non-OCR documents from the Anthology, except those documents that overlap with the HOO 2011 data.[20] Section headers, references, etc. are automatically removed. The Web 1T 5-gram corpus (Brants and Franz, 2006) is used for language modeling and collecting web N-gram counts. Table 7.2 gives an overview of the data sets.

HOO 2012

For the experiments on the HOO 2012 data, we use the HOO 2012 training data with the same data splits as in the previous chapter. 100 randomly selected documents from the HOO 2012 training data are used as the development set (HOO2012-DEV) and another 100 disjoint documents are randomly selected and used as the held-out development test set (HOO2012-DEVTEST). We train classifiers on the 1,044 documents of the FCE data set (FCE(1044)), tune parameters on HOO2012-DEV, and test on HOO2012-DEVTEST. The official HOO 2012 test data had to be deleted right after the HOO 2012 shared task and was not available for these experiments. All data sets for HOO 2011 and HOO 2012 are sentence segmented and tokenized. For the HOO 2012 data sets, additional case normalization is carried out using a standard SMT re-casing model that translates an un-cased sentence to a mixed-case sentence. As before, we use the Google

[20]In the previous chapter, we left out the complete 2010 ACL conference and workshops. In this chapter, we only excluded the exact documents that appear in the HOO 2011 test data.

Web 1T 5-gram corpus (Brants and Franz, 2006) for language modeling and collecting N-gram counts, the PukWaC corpus from the WaCky project (Baroni et al., 2009) for collecting web-scale dependency N-gram counts, and the New York Times section of the Gigaword corpus[21] and all normal-cased documents in the HOO 2012 training data for training the re-casing model.

7.3.2 Evaluation

We evaluate performance by computing precision, recall, and F_1 correction score as defined in Chapter 3 and the official HOO 2011 and HOO 2012 reports (Dale and Kilgarriff, 2011; Dale et al., 2012). F_1 correction score is simply the F_1 score (van Rijsbergen, 1979) between the corrections (called *edits* in HOO) proposed by a system and the gold-standard corrections. For the HOO 2011 shared task, F_1 correction score is computed "without bonus" which means that no credit is given for leaving optional corrections unchanged.

As described in Chapter 3, the evaluation of grammatical error correction is complicated by the fact that the set of system edits between the test sentences and the system outputs is ambiguous. The *MaxMatch* (M^2) scorer presented in Chapter 3 overcomes this problem through an efficient algorithm that computes the set of system edits which has the maximum overlap with the gold-standard edits. We use the M^2 scorer as the main evaluation metric in the HOO 2011 experiments. Additionally, we also evaluate with the official HOO 2011 scorer. For the HOO 2012, the shared task organizers gave specific instructions on how to extract the system edits for determiners and prepositions. As the task only evaluated these two error categories, computing the system edits was less ambiguous. We found that the M^2 scorer and the official HOO 2012 scorer resulted in the same scores in our HOO 2012 experiments. We note that all scorers (the M^2 scorer and the two HOO scorers) adhere to the same score definition and only differ in the way the system edits are computed. For statistical significance testing, we use sign-test with bootstrap re-sampling (Koehn, 2004) with 1,000 samples.

[21]LDC2009T13

7.3.3 SMT Baseline

For the HOO 2011 experiments, we build a baseline error correction system using the MOSES SMT system (Koehn et al., 2007). Word alignments are created automatically on "good-bad" parallel text from HOO2011-TRAIN using GIZA++ (Och and Ney, 2003), followed by phrase extraction using the standard heuristic (Koehn et al., 2003). The maximum phrase length is 5. Parameter tuning is done on the HOO2011-TUNE data with the PRO algorithm (Hopkins and May, 2011) implemented in MOSES. The optimization objective is sentence-level BLEU (Lin and Och, 2004). We note that the objective function is not the same as the final evaluation F_1 score. Also, the training and tuning data are small by SMT standards. The aim for the SMT baseline is not to achieve a state-of-the-art system, but to serve as the simplest possible baseline that uses only off-the-shelf software.

7.3.4 Pipeline Baseline

The second baseline system for the HOO 2011 experiments is an improved version of the pipeline of classifier-based and rule-based correction steps presented in the last chapter. Each step takes sentence segmented plain text as input, corrects one particular error category, and feeds the corrected text into the next step. No search or global inference is applied. The correction steps are:

1. Spelling correction
2. Article correction
3. Replacement preposition correction
4. Punctuation correction
5. Noun number correction

The HOO 2012 pipeline is the same as the one presented in the last chapter. The pipeline is briefly described here again for completeness and easy reference. The pipeline contains steps for spelling correction, article correction, preposition replacement correction, and preposition insertion and deletion correction. It does not contain steps for punctuation and noun number correction which are not evaluated in the HOO 2012 shared task.

1. Spelling correction
2. Article correction
3. Replacement preposition correction
4. Missing and unwanted preposition correction

At the end of every correction step, all proposed corrections are filtered using a 5-gram language model from the Web 1T 5-gram corpus and only corrections that strictly increase the normalized language model score of the sentence are applied.

We use the same NLP tools as before (repeated here for completeness): OpenNLP[22] for POS tagging, YamCha (Kudo and Matsumoto, 2003) for constituent chunking, and the MALT parser (Nivre et al., 2007) for dependency parsing. For language modeling, we use RandLM (Talbot and Osborne, 2007). For spelling correction, we use GNU Aspell[23].

Article Correction

As described in the previous chapter, correction is cast as a multi-class classification problem where a classifier tries to predict the correct word from a confusion set of possible choices. As the learning algorithm, we choose multi-class confidence-weighted (CW) learning (Crammer et al., 2009) which performed well in the experiments in the last chapter. For article correction, the possible classes are the articles *a*, *the*, and the null article ϵ. The article *an* is normalized as *a* and restored later using a rule-based heuristic. We consider all NPs that are not pronouns and do not have a non-article determiner, e.g., *this, that*. We use the features developed in the HOO 2012

[22]`http://opennlp.sourceforge.net`
[23]`http://aspell.net`

experiments in the previous chapter which include lexical and POS N-grams, lexical head words (Rozovskaya et al., 2011), web-scale N-gram count features from the Web 1T 5-gram corpus following (Bergsma et al., 2009), and dependency head and child features. During testing, a correction is proposed if the predicted article is different from the observed article used by the writer and the difference between the confidence score for the predicted article and the confidence score for the observed article is larger than a threshold. Threshold parameters are tuned via a grid-search on the development data. We tune a separate threshold value for each class.

For the HOO 2011 experiments, the article classifier is trained on over 5 million instances from ACL-ANTHOLOGY. As the training data is non-learner text that is not annotated with corrections, the article used by the writer cannot be used as a feature. The threshold parameters are tuned on HOO2011-TUNE. For the HOO 2012 experiments, the article classifier is trained on 79,000 instances from the FCE(1044) data set. As the FCE data set is learner text that is annotated with corrections, the article used by the writer is used as a feature.

Preposition Replacement Correction

Preposition replacement correction is analogous to article correction. It differs only in terms of the classes and the features. For preposition correction, the classes are 36 frequent English prepositions[24]. The features are surrounding lexical N-grams, web-scale N-gram counts, and dependency features which are the same as the features for the HOO 2012 experiments described in the previous chapter. For the HOO 2011 experiments, the preposition classifier is trained on 1 million training examples from the ACL-ANTHOLOGY. For the HOO 2012 experiments, the preposition classifier is trained on 34 thousand instances from the FCE(1044) data set. The preposition used by the writer is only used as a feature for the HOO 2012 experiments.

[24]*about, along, among, around, as, at, beside, besides, between, by, down, during, except, for, from, in, inside, into, of, off, on, onto, outside, over, through, to, toward, towards, under, underneath, until, up, upon, with, within, without*

Noun Number Correction

For noun number correction, the classes are *singular* and *plural*. The features are lexical N-grams, web-scale N-gram counts, dependency features, the noun lemma, and a binary countability feature. The noun number classifier is trained on over 5 million examples from ACL-ANTHOLOGY. During testing, the singular or plural word surface form is generated using WordNet (Fellbaum, 1998) and simple heuristics. Noun number correction is only done for the HOO 2011 experiments.

Punctuation Correction

Punctuation correction is done using a set of simple rules developed on the HOO 2011 development data. Punctuation correction is only done for the HOO 2011 experiments.

Missing and Unwanted Preposition Correction

Missing and unwanted preposition correction use the same binary classifiers described in the last chapter. For missing preposition correction, a binary classifier predicts whether a particular preposition p should be inserted before a noun phrase. Similarly, unwanted preposition correction is performed by binary classifiers that predict whether a particular preposition p in the text should be deleted. In both cases, the confusion set consists only of the preposition p and the "null preposition". A separate binary classifier is trained for each of the following seven prepositions: *about, at, for, in, of, on*, and *to*. The features for missing and unwanted preposition correction are the same as described in the previous chapter. For missing preposition correction, the features include surrounding lexical and POS N-gram features, head word features, and web-scale N-gram counts. For unwanted preposition correction, we only use web-scale N-gram count features. Missing and unwanted preposition correction are only done for the HOO 2012 experiments.

7.3.5 Decoder

We experiment with different decoder configurations with different proposers and expert models. For the HOO 2011 experiments, the simplest configuration of the decoder only has the spelling proposer and the language model expert. We then add the article proposer and expert, the preposition replacement proposer and expert, the punctuation proposer, and finally the noun number proposer and expert. We refer to the final configuration with all proposers and experts as the *full HOO 2011 decoder model*. For the HOO 2012 experiments, we start with a decoder that only performs spelling and article correction. Then we add preposition replacement correction, and finally missing and unwanted preposition correction. We refer to this configuration as the *full HOO 2012 decoder model*. Note that error categories are always corrected jointly and not in sequential steps as in the pipeline.

To make the results directly comparable to the pipeline, the decoder uses the same resources as the pipeline. As the expert models, we use a 5-gram language model from the Web 1T 5-gram corpus with the Berkeley LM (Pauls and Klein, 2011)[25] in the decoder and the CW-classifiers described in the last section for the HOO 2011 and HOO 2012 experiments, respectively. The spelling proposer uses the same spellchecker as the pipeline, and the punctuation proposer uses the same rules as the pipeline. The beam width is set to 10. The maximum number of iterations is set to 10 for the HOO 2011 experiments and to 3 for the HOO 2012 experiments. In earlier experiments, we found that larger values had no effect on the result. The simulated annealing temperature T is initialized to 10 and the exponential cooling schedule c is set to 0.9. The decoder weight vector is initialized as follows. The weight for the language model score and the weights for the classifier expert average scores are initialized to 1.0, and the weights for the classifier expert delta scores are initialized to -1.0. The weights for the correction count features are initialized to zero.

For PRO optimization, we use the HOO2011-TUNE and HOO2012-DEV data, respectively. We use the default PRO parameters from (Hopkins and May, 2011): we

[25]Berkeley LM is written in Java and was easier to integrate into the Java-based decoder than RandLM.

sample 5,000 hypothesis pairs from the N-best list (N = 100) for every input sentence and keep the top 50 sample pairs with the highest difference in F_1 score. The weights are optimized using MegaM (Daumé III, 2004) and interpolated with the previous weight vector with an interpolation parameter of 0.1. We normalize feature values to a unit interval. We use an early stopping criterion that terminates PRO if the objective function on the tuning data drops. For the HOO 2011 data, the tuning data is highly skewed as samples without errors greatly outnumber samples with errors. To balance the tuning data, we give a higher weight to sample pairs where the decoder proposed a valid correction. We found a weight of 20 to work well, based on initial experiments on the HOO2011-TUNE data. For the HOO 2012 data, we found that re-weighting the samples was not necessary. Therefore, we do not re-weight the samples in the HOO 2012 experiments. We keep all these parameters fixed for all experiments.

7.4 Results

The complete results of the HOO 2011 and HOO 2012 experiments are shown in Table 7.3 and Table 7.4, respectively. Each row contains the results for one error correction system.

HOO 2011

For the HOO 2011 experiments, each system is scored on the original and official gold-standard annotations, both with the M^2 scorer and the official HOO 2011 scorer. This results in four sets of precision, recall, and F_1 scores for each system. The best result in the H00 2011 shared task was achieved by the UI Run1 system by Rozovskaya *et al.* (2011). We include their system as a reference point.

We make the following observations. First, the scores on the official gold-standard annotations are higher compared to the original gold-standard annotations. We note that the gap between the two annotations is the largest for the UI Run1 system which confirms the suspected bias of the official gold-standard annotations in favor of participating systems. Second, the scores computed with the M^2 scorer are higher than the scores computed with the official HOO 2011 scorer. With more error categories and more ambiguity in the edits segmentation, the gap between the scorers widens. In the case of the full HOO 2011 pipeline and decoder model, the HOO 2011 scorer even shows a *decrease* in F_1 score when the score actually goes up as shown by the M^2 scorer. We therefore focus on the scores of the M^2 scorer from now on. The SMT baseline achieves 8.68% and 11.21% F_1 score on the original and official gold standard, respectively. Although the worst system in our experiments, it would still have claimed the third place in the HOO 2011 shared task. One problem is certainly the small amount of training data. Another reason is that the phrase-based model is unaware of syntactic structure and cannot express correction rules of the form $NP \rightarrow the\ NP$. Instead, it has to have seen the exact correction rule, e.g., $house \rightarrow the\ house$, in the training data. As a result, the model does not generalize well. The pipeline achieves state-of-the-art results. Each additional correction step improves the score. Our proposed decoder achieves the best result. When only a few error categories are corrected, the pipeline and the decoder are close to each other. When more error categories are added, the gap between the pipeline and the decoder becomes larger. The full HOO 2011 decoder model achieves an F_1 score of 23.48% and 25.48% on the original and official gold standard, respectively, which is statistically significantly better than both the pipeline system and the UI Run1 system.

HOO 2012

For the HOO 2012 experiments, there is only one gold-standard annotation available for evaluation.[26] Evaluation is done with the M^2 scorer and the official HOO 2012 scorer which give the same result in this case. The pipeline experiments are the same as those presented in the last chapter on the HOO2012-DEVTEST data set. We repeat the results here for easy comparison with the decoder results. Each correction step in the pipeline improves the score. The decoder again achieves the best result, improving

[26]For the official HOO 2012 test set, there exist two gold standards (before and after revision) but not for the development test set.

over the pipeline in every experiment. Although the improvements over the pipeline are small, the improvements are statistically significant in all experiments. The full HOO 2012 decoder model achieves an F_1 score of 35.83% on the development test set. With this, the decoder improves over the state-of-the-art pipeline system that achieved the highest correction score in the HOO 2012 shared task. The consistent improvement of the decoder over the pipeline model on both data sets shows the advantage of the decoder method.

7.5 Discussion

As pointed out in Section 7.2.5, the majority of sentences require zero or few corrections. Therefore, the depth of the search tree is typically small. In our HOO 2011 experiments, for example, the average depth of the search tree is only 1.9 (i.e., 0.9 corrections per sentence) on the test set. On the other hand, there are many possible hypotheses that can be proposed for any sentence. The breath of the search tree is therefore quite large. In our HOO 2011 experiments, the decoder explored on average 99 hypotheses per sentence on the test set.

We found that PRO tuning is very important to achieve good performance for the decoder. Most importantly, PRO tunes the correction count features that bias the decoder against over-correcting sentences thus improving precision. But PRO is also able to improve recall during tuning. Table 7.5 shows the trajectory of the performance for the full HOO 2011 decoder model during PRO tuning on HOO2011-TUNE. After PRO tuning has converged, we inspect the learned weight vector and observe some interpretable patterns learned by PRO. First, the language model score and all classifier expert average scores receive positive weights, while all classifier expert delta scores receive negative weights, in line with the initial intuition described in Section 7.2.3. Second, most correction count features receive negative weights, thus acting as a bias against correction if it is not necessary. Finally, the correction count features reveal which corrections are more likely and which are less likely. For example, article replacement errors are less

System	**M^2 scorer**			**HOO scorer**		
	P	R	F_1	P	R	F_1
UI Run1	40.86	11.21	**17.59**	38.13	10.42	**16.37**
	P	R	F_1	P	R	F_1
SMT	9.84	7.77	**8.68**	15.25	5.31	**7.87**
Pipeline	P	R	F_1	P	R	F_1
Spelling	50.00	0.79	1.55	40.00	0.64	1.25
+ Articles	30.86	10.23	15.36	28.04	9.55	14.25
+ Preposition replacement	27.44	11.90	16.60	24.82	11.15	15.38
+ Punctuation	28.91	14.55	19.36 †	26.57	13.91	**18.25** †
+ Noun number	28.77	16.13	**20.67** †	24.68	14.22	18.04 †
Decoder	P	R	F_1	P	R	F_1
Spelling	36.84	0.69	1.35	22.22	0.41	0.80
+ Articles	19.84	12.59	15.40	17.99	12.00	14.39
+ Preposition replacement	22.62	14.26	17.49 $*$	19.30	12.95	15.50
+ Punctuation	24.27	18.09	20.73 $*$†	20.40	16.24	18.08
+ Noun number	30.28	19.17	**23.48** $*$†	24.29	16.24	**19.46** $*$†

(a) Original gold standard

System	**M^2 scorer**			**HOO scorer**		
	P	R	F_1	P	R	F_1
UI Run1	54.61	14.57	**23.00**	50.72	13.34	**21.12**
	P	R	F_1	P	R	F_1
SMT	23.35	7.38	**11.21**	15.82	5.30	**7.93**
Pipeline	P	R	F_1	P	R	F_1
Spelling	50.00	0.76	1.49	40.00	0.61	1.20
+ Articles	34.42	10.97	16.64	31.78	10.41	15.68
+ Preposition replacement	30.54	12.77	18.01	27.90	12.04	16.82
+ Punctuation	32.88	15.99	21.51	30.63	15.41	**20.50**
+ Noun number	32.34	17.50	**22.71**	28.36	15.71	20.22
Decoder	P	R	F_1	P	R	F_1
Spelling	36.84	0.66	1.30	22.22	0.42	0.83
+ Articles	22.45	13.72	17.03 $*$	20.70	13.27	16.16
+ Preposition replacement	24.84	15.14	18.81 $*$	21.36	13.78	16.74
+ Punctuation	27.13	19.58	22.75 $*$	23.07	17.65	19.99
+ Noun number	33.59	20.53	**25.48** $*$†	27.30	17.55	**21.36** $*$

(b) Official gold standard

Table 7.3: Experimental results on HOO2011-TEST. Precision, recall, and F_1 score are shown in percent. The best F_1 score for each system is highlighted in bold. Statistically significant improvements ($p < 0.01$) over the pipeline baseline are marked with an asterisk ($*$). Statistically significant improvements over the UI Run1 system are marked with a dagger (†). All improvements of the pipeline and the decoder over the SMT baseline are statistically significant.

Pipeline	P	R	F_1
Articles	54.10	11.01	18.30
+ Preposition replacement (RT)	57.38	19.97	29.63
+ Missing/unwanted preposition (MT/UT)	54.84	26.12	**35.39**
Decoder	P	R	F_1
Articles	36.22	17.16	23.28 $*$
+ Preposition replacement (RT)	46.86	22.92	30.78 $*$
+ Missing/unwanted preposition (MT/UT)	45.73	29.45	**35.83** $*$

Table 7.4: Experimental results on HOO2012-DEVTEST. Precision, recall, and F_1 score are shown in percent. The best F_1 score for each system is highlighted in bold. Statistically significant improvements ($p < 0.01$) over the pipeline baseline are marked with an asterisk ($*$).

PRO iteration	P	R	F_1
1	14.13	20.17	16.62
2	19.71	20.85	20.27
3	23.12	21.03	22.02
4	24.35	20.85	22.47
5	25.53	20.51	22.75
6	26.27	20.34	22.93
7	27.25	20.68	**23.52**
8	26.73	19.83	22.77

Table 7.5: PRO tuning of the full HOO 2011 decoder model on HOO2011-TUNE

common in the HOO2011-TUNE data than article insertions or deletions. The weights learned for the article correction count features shown in Table 7.6 reflect this.

In the HOO 2012 experiments, the improvements of the decoder over the pipeline were smaller than in the HOO 2011 experiments. We believe that this is a result of the restriction of the HOO 2012 task to only article and preposition errors. Articles and prepositions interact less with each other than, for example, articles and noun number. As the decoder's strength lies in correcting sentences with multiple interacting errors, the relative improvements over the pipeline baseline are smaller when fewer and less

Feature	Weight
$a \rightarrow the$	-1.3660
$a \rightarrow \epsilon$	0.5253
$the \rightarrow a$	-0.9997
$the \rightarrow \epsilon$	0.0532
$\epsilon \rightarrow a$	0.0694
$\epsilon \rightarrow the$	-0.0529

Table 7.6: Example of PRO-tuned weights for article correction count features for the full HOO 2011 decoder model.

interacting error categories are chosen.

Although the decoder achieves state-of-the-art results, there remain many error categories which the decoder currently cannot correct. This includes, for example, verb form errors (*Much research (have $\rightarrow$ has) been put into ...*) and lexical choice errors (*The (concerned $\rightarrow$ relevant) relation ...*). We believe that our decoder provides a promising framework to build grammatical error correction systems that include these types of errors in the future.

7.6 Conclusion

We have presented a novel beam-search decoder for grammatical error correction. The model performs end-to-end correction of whole sentences with multiple, interacting errors, is discriminatively trained, and incorporates existing classifier-based models for error correction. Our decoder achieves an F_1 correction score of 25.48% on the HOO 2011 test set and an F_1 correction score of 35.83% on the HOO 2012 development test set which outperforms the current state of the art on both data sets.

Chapter 8

Conclusion

In this thesis, we have made several contributions that advance grammatical error correction research. We started by motivating the need for automatic grammatical error correction systems and why we believe that computers can achieve this goal. Next, we presented the NUS Corpus of Learner English (NUCLE), a fully annotated one-million word corpus of learner text which was built as part of this thesis. We hope that this corpus will be a useful resource for grammatical error correction research in the future. We have presented a novel method, called MaxMatch (M^2), for evaluating grammatical error correction that overcomes problems in current evaluation tools. In Chapter 4, we presented a novel approach for training classifiers for grammatical error correction based on Alternating Structure Optimization. Experiments for article and preposition errors show the advantage of the ASO approach over two baseline methods and two commercial grammar checking software packages. In Chapter 5, we presented a novel approach for correcting lexical choice errors. Our approach exploits the semantic similarity of words in the writer's native language based on paraphrases extracted from a parallel corpus. Experiments on real-world learner data have shown that our approach outperforms traditional approaches based on edit distance, homophones, and synonyms by a large margin. In Chapter 6, we presented a pipeline architecture for end-to-end grammatical error correction systems. The NUS system submissions based on this architecture achieved the second highest correction F_1 score in the HOO 2011 shared task and the highest correction F_1 score in the HOO 2012 shared task. Finally, we presented a novel beam-search decoder for grammatical error correction. The model performs end-to-end correction of whole sentences with multiple, interacting errors, is discriminatively trained, and incorporates existing classifier-based models for error correction. The architecture of the decoder provides a new framework for how to build grammatical error correction systems. Our decoder outperforms the state-of-the-art pipeline approach on both the HOO 2011 and HOO 2012 shared task data.

While this thesis has advanced the current state of the art for grammatical error correction in several directions, grammatical error correction is still an emerging research topic in natural language processing and much work remains to be done. For example, most grammatical error correction research, including this thesis, restricts the context of a grammatical error to a single sentence. It is obvious that certain types of grammatical errors, like co-reference and discourse, have a scope beyond a single sentence. Extensions of existing grammatical error correction models to paragraph and document contexts are needed to correct these types of errors. In addition, grammatical error correction systems are currently not able to say *why* something is an error and are not able to justify their proposed corrections. If an algorithm could provide feedback to a language learner as to why a particular word has to be used in that particular context, it would increase trust in the system and enhance the learning experience of the learner.

In addition, the performance of current grammar correction systems still needs to be improved further. While the methods presented in this thesis have shown state-of-the-art performance, the final F_1 scores for the decoder model, for example, are only in the 20% - 30% range, which still appears low in absolute terms. This raises the question how much nearer this thesis has brought us to the vision of practical grammar correction systems for language learners. My answer to this question would be that we are probably closer to seeing practical grammar correction systems than the numbers might suggest. First, the upper bound for the grammar correction task is not 100% F_1 score. We have shown in Chapter 3 that grammatical error correction is a difficult task where even trained annotators have problems to achieve good agreement. The upper bound for grammar correction systems should therefore be the average F_1 score of a human annotator measured against the gold standard, which I believe would be considerably lower than 100%. Future work is needed to investigate the human annotator agreement issue and to quantify the upper bound for automatic error correction. Second, we have shown in Chapter 4 that our classifiers already outperform commercial grammar checking software. In other words, a practical system built on the results of this thesis would already provide more accurate corrections than the existing solutions in the market. Finally, I believe that grammatical error correction techniques will be used to assist humans in tasks like proofreading and text editing, rather than outright replacing them. Just like machine translation is not perfect but it is often good enough to get a first translation for post-editing, grammatical error correction systems could be used to automatically scan through a text and make the first round of corrections which would then be examined by a human editor. Despite these remaining obstacles, it is encouraging that during the time that this thesis was done, we could see that interest in grammatical error correction research was clearly picking up and that research systems approach the level of accuracy where they start to become useful for practical applications where they can improve people's lives.

Bibliography

[Ando and Zhang2005] R.K. Ando and T. Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

[Bannard and Callison-Burch2005] C. Bannard and C. Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*, pages 597–604.

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

[Bergsma et al.2009] S. Bergsma, D. Lin, and R. Goebel. 2009. Web-scale N-gram models for lexical disambiguation. In *Proceedings of IJCAI*, pages 1507–1512.

[Bergsma et al.2010] S. Bergsma, E. Pitler, and D. Lin. 2010. Creating robust supervised classifiers via web-scale N-gram data. In *Proceedings of ACL*, pages 865–874.

[Bird et al.2008] S. Bird, R. Dale, B.J. Dorr, B. Gibson, M. Joseph, M.Y. Kan, D. Lee, B. Powley, D.R. Radev, and Y.F. Tan. 2008. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of LREC*, pages 1755–1759.

[Brants and Franz2006] T. Brants and A. Franz. 2006. Web 1T 5-gram corpus version 1.1. Technical report, Google Research.

[Brants et al.2007] T. Brants, A.C. Popat, P. Xu, F. J. Och, and J. Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP*, pages 858–867.

[Brockett et al.2006] C. Brockett, W.B. Dolan, and M. Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of ACL*, pages 249–256.

[Callison-Burch et al.2012] C. Callison-Burch, P. Koehn, C. Monz, M. Post, R. Soricut, and L. Specia. 2012. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of WMT*, pages 10–51.

[Carlson et al.2001] A.J. Carlson, J. Rosen, and D. Roth. 2001. Scaling up context-sensitive text correction. In *Proceedings of IAAI*, pages 45–50.

[Chan and Ng2005] Y.S. Chan and H. T. Ng. 2005. Scaling up word sense disambiguation via parallel texts. In *Proceedings of AAAI*, pages 1037–1042.

[Chang et al.2008] Y.C. Chang, J. S. Chang, H.J. Chen, and H.C. Liou. 2008. An automatic collocation writing assistant for Taiwanese EFL learners: A case of corpus-based NLP technology. *Computer Assisted Language Learning*, 21(3):283–299.

[Chodorow et al.2007] M. Chodorow, J. Tetreault, and N.R. Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions*, pages 25–30.

[Clark and Curran2007] S. Clark and J.R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

[Cohen1960] J. Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

[Cormen et al.2001] T. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. 2001. *Introduction to Algorithms*. MIT Press, Cambridge, MA.

[Crammer et al.2009] K. Crammer, M. Dredze, and A. Kulesza. 2009. Multi-class confidence weighted algorithms. In *Proceedings of EMNLP*, pages 496–504.

[Dahlmeier and Ng2011a] D. Dahlmeier and H.T. Ng. 2011a. Correcting semantic collocation errors with L1-induced paraphrases. In *Proceedings of EMNLP*, pages 107–117.

[Dahlmeier and Ng2011b] D. Dahlmeier and H.T. Ng. 2011b. Grammatical error correction with alternating structure optimization. In *Proceedings of ACL:HLT*, pages 915–923.

[Dahlmeier and Ng2012a] D. Dahlmeier and H.T. Ng. 2012a. A beam-search decoder for grammatical error correction. In *Proceedings of EMNLP*, pages 568–578.

[Dahlmeier and Ng2012b] D. Dahlmeier and H.T. Ng. 2012b. Better evaluation for grammatical error correction. In *Proceedings of HLT-NAACL*, pages 568–572.

[Dahlmeier et al.2011] D. Dahlmeier, H. T. Ng, and T. P. Tran. 2011. NUS at the HOO 2011 pilot shared task. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 257–259.

[Dahlmeier et al.2012] D. Dahlmeier, H. T. Ng, and E. J. F. Ng. 2012. NUS at the HOO 2012 shared task. In *Proceedings of the Seventh Workshop on Innovative Use of NLP for Building Educational Applications*, pages 216–224.

[Dale and Kilgarriff2011] R. Dale and A. Kilgarriff. 2011. Helping Our Own: The HOO 2011 pilot shared task. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 242–249.

[Dale et al.2012] R. Dale, I. Anisimoff, and G. Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Innovative Use of NLP for Building Educational Applications*, pages 54–62.

[Daumé III2004] H. Daumé III. 2004. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`.

[De Felice2008] R. De Felice. 2008. *Automatic Error Detection in Non-native English*. Ph.D. thesis, St Catherine's College, University of Oxford.

[Désilets and Hermet2009] A. Désilets and M. Hermet. 2009. Using automatic roundtrip translation to repair general errors in second language writing. In *Proceedings of MT-Summit XII*.

[Dredze et al.2008] M. Dredze, K. Crammer, and F. Pereira. 2008. Confidence-weighted linear classification. In *Proceedings of ICML*, pages 184–191.

[Dyer et al.2008] C. Dyer, S. Muresan, and P. Resnik. 2008. Generalizing word lattice translation. In *Proceedings of ACL:HLT*, pages 1012–1020.

[Farghal and Obiedat1995] M. Farghal and H. Obiedat. 1995. Collocations: A neglected variable in EFL. *International Review of Appplied Linguistics*, 33(4):315–31.

[Fellbaum1998] C. Fellbaum, editor. 1998. *WordNet: An electronic lexical database*. MIT Press, Cambridge,MA.

[Firth1957] J.R. Firth. 1957. *Papers in Linguistics 1934-1951*. Oxford University Press, London.

[Foster et al.2006] G. Foster, R. Kuhn, and H. Johnson. 2006. Phrasetable smoothing for statistical machine translation. In *Proceedings of EMNLP*, pages 53–61.

[Foster2007] J. Foster. 2007. Treebanks gone bad: parser evaluation and retraining using a treebank of ungrammatical sentences. *International Journal on Document Analysis and Recognition*, 10(3-4):129–207.

[Freund and Schapire1999] Y. Freund and R.E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

[Futagi et al.2008] Y. Futagi, P. Deane, M. Chodorow, and J. Tetreault. 2008. A computational approach to detecting collocation errors in the writing of non-native speakers of English. *Journal of Computer-Assisted Learning*, 21:353–367.

[Gale et al.1992] W. Gale, K Church, and D. Yarowsky. 1992. Work on statistical methods for word sense disambiguation. In *Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language*, pages 54–60.

[Gamon et al.2008] M. Gamon, J. Gao, C. Brockett, A. Klementiev, W.B. Dolan, D. Belenko, and L. Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP*, pages 449–456.

[Gamon2010] M. Gamon. 2010. Using mostly native data to correct errors in learners' writing: A meta-classifier approach. In *Proceedings of HLT-NAACL*, pages 163–171.

[Golding and Roth1999] A.R. Golding and D. Roth. 1999. A winnow-based approach to context-sensitive spelling correction. *Machine Learning*, 34:107–130.

[Golding1995] A.R. Golding. 1995. A Bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Third Workshop on Very Large Corpora*, pages 39–53.

[Graddol2006] D. Graddol. 2006. *English Next*. The English Company.

[Granger et al.2002] S. Granger, F. Dagneaux, E. Meunier, and M. Paquot. 2002. *The International Corpus of Learner English*. Presses Universitaires de Louvain, Louvain-la-Neuve, Belgium.

[Hagen1995] K.L. Hagen. 1995. Unification-based parsing applications for intelligent foreign language tutoring systems. *Calico Journal*, 2(2):2–8.

[Haghighi et al.2009] A. Haghighi, J. Blitzer, J. DeNero, and D. Klein. 2009. Better word alignments with supervised ITG models. In *Proceedings of ACL-IJCNLP*, pages 923–931.

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

[Han et al.2010] N.R. Han, J. Tetreault, S.H. Lee, and J.Y. Ha. 2010. Using an error-annotated learner corpus to develop an ESL/EFL error correction system. In *Proceedings of LREC*, pages 763–770.

[Heidorn et al.1982] G.E. Heidorn, K. Jensen, L.A. Miller, R.J. Byrd, and M. Chodorow. 1982. The Epistle text-critiquing system. *IBM Systems Journal*, 21(3):305–326.

[Heidorn2000] G.E Heidorn, 2000. *Intelligent writing assistance*, pages 181–207. Handbook of Natural Language Processing. Marcel Dekker, New York.

[Heift and Schulze2007] Trude Heift and Mathias Schulze. 2007. *Errors and Intelligence in Computer-Assisted Language Learning*. Routledge, London, UK.

[Hopkins and May2011] M. Hopkins and J. May. 2011. Tuning as ranking. In *Proceedings of EMNLP*, pages 1352–1362.

[Izumi et al.2003] E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *Companion Volume to the Proceedings of ACL*, pages 145–148.

[Jelinek1998] F. Jelinek. 1998. *Statistical methods for speech recognition*. MIT press, Cambridge, MA.

[Klein and Manning2003a] D. Klein and C.D. Manning. 2003a. Accurate unlexicalized parsing. In *Proceedings of ACL*, pages 423–430.

[Klein and Manning2003b] D. Klein and C.D. Manning. 2003b. Fast exact inference with a factored model for natural language processing. *Advances in Neural Information Processing Systems (NIPS 2002)*, 15:3–10.

[Knight and Chander1994] K. Knight and I. Chander. 1994. Automated postediting of documents. In *Proceedings of AAAI*, pages 779–784.

[Koehn et al.2003] P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*, pages 48–54.

[Koehn et al.2007] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Companion Volume to the Proceedings of ACL Demo and Poster Sessions*, pages 177–180.

[Koehn2004] P. Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

[Koehn2006] Philipp Koehn. 2006. Statistical machine translation: the basic, the novel, and the speculative. Tutorial at EACL.

[Koehn2010] P. Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, Cambridge, UK.

[Kudo and Matsumoto2003] T. Kudo and Y. Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of ACL*, pages 24–31.

[Landis and Koch1977] J.R. Landis and G.G Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

[Lapata and Keller2005] M. Lapata and F. Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31.

[Leacock et al.2010] C. Leacock, M. Chodorow, M. Gamon, and J. Tetreault. 2010. *Automated Grammatical Error Detection for Language Learners*. Morgan & Claypool Publishers.

[Lee and Knutsson2008] J. Lee and O. Knutsson. 2008. The role of PP attachment in preposition generation. In *Proceedings of CICLing*, pages 643–654.

[Lee and Ng2002] Y.K. Lee and H.T. Ng. 2002. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of EMNLP*, pages 41–48.

[Lee and Seneff2006] J. Lee and S. Seneff. 2006. Automatic grammar correction for second-language learners. In *Proceedings of Interspeech*, pages 1978–1981.

[Lee2004] J. Lee. 2004. Automatic article restoration. In *Proceedings of HLT-NAACL*, pages 31–36.

[Levenshtein1966] V. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710.

[Liang et al.2006] P. Liang, B. Taskar, and D. Klein. 2006. Alignment by agreement. In *Proceedings of HLT-NAACL*, pages 104–111.

[Lin and Och2004] C.-Y. Lin and F.J. Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of COLING*, pages 501–507.

[Liu and Ng2007] C. Liu and H. T. Ng. 2007. Learning predictive structures for semantic role labeling of NomBank. In *Proceedings of ACL*, pages 208–215.

[Liu et al.2009] A.L. Liu, D. Wible, and N.L. Tsao. 2009. Automated suggestions for miscollocations. In *Proceedings of the ACL 4th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 47–50.

[Liu et al.2010a] C. Liu, D. Dahlmeier, and H.T. Ng. 2010a. PEM: a paraphrase evaluation metric exploiting parallel texts. In *Proceedings of EMNLP*, pages 923–932.

[Liu et al.2010b] C. Liu, D. Dahlmeier, and H.T. Ng. 2010b. TESLA: Translation evaluation of sentences with linear-programming-based analysis. In *Proceedings of WMT and MetricsMATR*, pages 354–359.

[Low et al.2005] J.K. Low, H.T. Ng, and W. Guo. 2005. A maximum entropy approach to Chinese word segmentation. In *Proceedings of the 4th SIGHAN Workshop*, pages 161–164.

[MacDonald et al.1982] N.H. MacDonald, L.T. Frase, P.S. Gingrich, and S.A. Keenan. 1982. The writer's workbench: Computer aids for text analysis. *IEEE Transactions on Communications*, 30(1):105–110.

[Madnani and Dorr2010] N. Madnani and B.J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

[Marcus et al.1993] M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

[McCarthy and Navigli2007] D. McCarthy and R. Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53.

[Meng2008] J. Meng. 2008. Erroneous collocations caused by language transfer in Chinese EFL writing. *US-China Foreign Language*, 6:57–61.

[Minnen et al.2000] G. Minnen, F. Bond, and A. Copestake. 2000. Memory-based learning for article generation. In *Proceedings of CoNLL*, pages 43–48.

[Mitton1992] R. Mitton. 1992. A description of a computer-usable dictionary file based on the Oxford Advanced Learner's Dictionary of Current English.

[Munson et al.2012] T. Munson, J. Sarich, S. Wild, S. Benson, and L.C. McInnes. 2012. Tao 2.0 users manual. Technical Report ANL/MCS-TM-322, Mathematics and Computer Science Division, Argonne National Laboratory.

[Nagata et al.2006] R. Nagata, A. Kawai, K. Morihiro, and N. Isu. 2006. A feedback-augmented method for detecting errors in the writing of learners of English. In *Proceedings of COLING-ACL*, pages 241–248.

[Ng and Chan2007] H.T. Ng and Y.S. Chan. 2007. SemEval-2007 task 11: English lexical sample task via English-Chinese parallel text. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 54–58.

[Ng and Lee1996] H.T. Ng and H.B. Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense: an examplar-based approach. In *Proceedings of ACL*, pages 40–47.

[Ng et al.2003] H.T. Ng, B. Wang, and Y.S. Chan. 2003. Exploiting parallel texts for word sense disambiguation: An empirical study. In *Proceedings of ACL*, pages 455–462.

[Ng et al.2013] H.T. Ng, S.M. Wu, Y. Wu, C. Hadiwinoto, and J. Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *To appear in Proceedings of the Seventeenth Conference on Computational Natural Language Learning*.

[Nicholls2003] D. Nicholls. 2003. The Cambridge learner corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 Conference*, pages 572–581.

[Nivre et al.2007] J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and M. Marsi. 2007. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

[Och and Ney2003] F.J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

[Och2003] F. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

[Pan and Yang2010] S.J. Pan and Q. Yang. 2010. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

[Papineni et al.2002] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318.

[Park and Levy2011] Y. A. Park and R. Levy. 2011. Automated whole sentence grammar correction using a noisy channel model. In *Proceedings of ACL:HLT*, pages 934–944.

[Pauls and Klein2011] A. Pauls and D. Klein. 2011. Faster and smaller N-gram language models. In *Proceedings of ACL:HLT*, pages 258–267.

[Rozovskaya and Roth2010a] A. Rozovskaya and D. Roth. 2010a. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP*, pages 961–970.

[Rozovskaya and Roth2010b] A. Rozovskaya and D. Roth. 2010b. Training paradigms for correcting errors in grammar and usage. In *Proceedings of HLT-NAACL*, pages 154–162.

[Rozovskaya et al.2011] A. Rozovskaya, M. Sammons, J. Gioja, and D. Roth. 2011. University of Illinois system in HOO text correction shared task. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 263–266.

[Russell and Norvig2010] S. Russell and P. Norvig, 2010. *Artificial Intelligence: A Modern Approach*, chapter 27. Prentice Hall, Upper Saddle River, NJ.

[Schneider and McCoy1998] D. Schneider and K. McCoy. 1998. Recognizing syntactic errors in the writing of second language learners. In *Proceedings of COLING-ACL*, pages 1198–1204.

[Schwind1990] C.B. Schwind. 1990. Feature grammars for semantic analysis. *Computer Intelligence*, 6:172–178.

[Shei and Pain2000] C.C. Shei and H. Pain. 2000. An ESL writer's collocational aid. *Computer Assisted Language Learning*, 13:167–182.

[Snover et al.2009] M. Snover, N. Madnani, B. Dorr, and R. Schwartz. 2009. Fluency, adequacy, or HTER? Exploring different human judgments with a tunable MT metric. In *Proceedings of WMT*, pages 259–268.

[Swan and Smith2001] M. Swan and B. Smith. 2001. *Learner English: A Teacher's Guide to Interference and Other Problems*. Cambridge University Press, Cambridge, UK.

[Talbot and Osborne2007] D. Talbot and M. Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proceedings of ACL*, pages 512–519.

[Tetreault and Chodorow2008a] J. Tetreault and M. Chodorow. 2008a. Native judgments of non-native usage: Experiments in preposition error detection. In *Proceedings of the Workshop on Human Judgements in Computational Linguistics*, pages 24–32.

[Tetreault and Chodorow2008b] J. Tetreault and M. Chodorow. 2008b. The ups and downs of preposition error detection in ESL writing. In *Proceedings of COLING*, pages 865–872.

[Tetreault et al.2010] J. Tetreault, J. Foster, and M. Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 353–358.

[van Rijsbergen1979] C. J. van Rijsbergen. 1979. *Information Retrieval*. Butterworth, Oxford, UK, 2nd edition.

[Wible et al.2003] D. Wible, C.H. Kuo, N.L. Tsao, A. Liu, and H.L. Lin. 2003. Bootstrapping in a language learning environment. *Journal of Computer-Assisted Learning*, 19:90–102.

[Wu and Zhou2003] H. Wu and M. Zhou. 2003. Synonymous collocation extraction using translation information. In *Proceedings of ACL*, pages 120–127.

[Wu et al.2010] J.C. Wu, Y.C. Chang, T. Mitamura, and J.S. Chang. 2010. Automatic collocation suggestion in academic writing. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 115–119.

[Yannakoudakis et al.2011] H. Yannakoudakis, T. Briscoe, and B. Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of ACL:HLT*, pages 180–189.

[Yarowsky1994] D. Yarowsky. 1994. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. In *Proceedings of ACL*, pages 88–95.

[Yi et al.2008] X. Yi, J. Gao, and W.B. Dolan. 2008. A web-based English proofing system for English as a second language users. In *Proceedings of IJCNLP*, pages 619–624.

[Zhong and Ng2009] Z. Zhong and H.T. Ng. 2009. Word sense disambiguation for all words without hard labor. In *Proceeding of IJCAI*, pages 1616–1621.