COMPUTATIONAL MODELS OF PROBLEMS WITH WRITING OF ENGLISH AS A SECOND LANGUAGE LEARNERS

by

Huichao Xue

B.S. in Computer Science, Fudan University, 2008

Submitted to the Graduate Faculty of

the Kenneth P. Dietrich School of Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2015

UNIVERSITY OF PITTSBURGH

KENNETH P. DIETRICH SCHOOL OF ARTS AND SCIENCES

This dissertation was presented

by

Huichao Xue

It was defended on

Sep 26, 2014

and approved by

Rebecca Hwa, PhD, Dietrich School of Arts and Sciences

Milos Hauskrecht, PhD, Dietrich School of Arts and Sciences

Janyce Wiebe, PhD, Dietrich School of Arts and Sciences

Joel Tetreault, PhD, Yahoo! Labs

Dissertation Director: Rebecca Hwa, PhD, Dietrich School of Arts and Sciences

COMPUTATIONAL MODELS OF PROBLEMS WITH WRITING OF ENGLISH AS A SECOND LANGUAGE LEARNERS

Huichao Xue, PhD

University of Pittsburgh, 2015

Learning a new language is a challenging endeavor. As a student attempts to master the grammar usage and mechanics of the new language, they make many mistakes. Detailed feedback and corrections from language tutors are invaluable to student learning, but it is time consuming to provide such feedback. In this thesis, I investigate the feasibility of building computer programs to help to reduce the efforts of English as a Second Language (ESL) tutors. Specifically, I consider three problems: (1) whether a program can identify areas that may need the tutor's attention, such as places where the learners have used redundant words; (2) whether a program can auto-complete a tutor's corrections by inferring the location and reason for the correction; (3) for detecting misusages of prepositions, a common ESL error type, whether a program can automatically construct a set of potential corrections by finding words that are more likely to be confused with each other (known as a *confusion set*).

The viability of these programs depends on whether aspects of the English language and common ESL mistakes can be described by computational models. For each task, building computational models faces unique challenges: (1) In highlighting redundant areas, it is difficult to precisely define "redundancy" in a computer's language. (2) In auto-completing tutors' annotations, it is difficult for computers to correctly interpret how many writing problems were addressed during revision. (3) In confusion set construction, it is difficult to infer which words are more likely confused with the given word. To address these challenges, this thesis presents different model alternatives for each task. Empirical experiments demonstrate the degrees of success to which computational models can help with detecting and correcting ESL writing problems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

PhD study is such a unique life experience. As this work comes to an end, I would like to bring my sincere thanks to people accompanied me throughout the journey. You made my journey joyful and complete.

I first thank my advisor Dr. Rebecca Hwa. You patiently guided me through the scientific discovery process. You taught me how to manage time and to communicate clearly. In moments of depression, it was your continuous encouragement that moved me forward. Your positive attitude toward challenges also benefits me beyond academic researches.

I extend my gratitude to my dissertation committee: Dr. Janyce Wiebe, Dr. Milos Hauskrecht and Dr. Joel Tetreault. Your valuable comments and suggestions during my study kept me on the right track. Also many thanks to my intern mentors Dr. Richard Zens, Dr. Su-Youn Yoon and Dr. Suma Bhat. The working experience with you broadened my sight, and equipped me with powerful tools.

Thank all members in Pitt NLP group. NLP is an exciting field. I was lucky to explore this field with you all. For the fun after-hour discussions, I specially thank my excellent colleagues Fan Zhang, Lingjia Deng and Huma Hashemi.

My final thanks to my friends and families, for all the joy and tear. I thank Huadong Wang, Yu Du, Miao Zhou, Yao Sun, Lei Jin, Yinglin Sun, Zitao Liu and Xiangmin Fan. We together enjoyed a colorful time. Finally I thank my wife Yafei Wei and my parents, for your never ending support.

1.0 INTRODUCTION

English as a Second Language (ESL) learners need feedback from their language tutors in order to learn to write better. However, providing high-quality feedback takes up a great deal of time and effort. This is challenging for tutors who have to instruct many students. Although there are more and more computer systems that attempt to assist human tutors with their tasks, these systems, too, are developed based on the knowledge of the human tutors; therefore, there is at least a one-time cost on some tutors' time and effort when these systems are being developed.

This thesis investigates methods for developing computational models of language tutors' feedback in order to reduce their efforts. My work addresses problems from two major categories: (1) how to model different types of tutors feedback for students; and (2) how to simulate tutor knowledge for training an automated system. In the first category, I tackle two problems: helping tutors to **detect redundant words or phrases** in student writings; and helping tutors to auto-complete their feedback by **inferring the rationale behind each correction** – that is, identify the locations and reasons of their edits. For the second category, I have developed models to facilitate the construction of an integral component of an automatic feedback generation system, called *confusion sets*, without requesting additional input from human tutors.

1.1 THESIS STATEMENT

Computational models of ESL errors can make language tutoring more efficient, both by helping to automate instructional feedback directly and by facilitating the development of such assistive systems without time-consuming annotation efforts from language experts.

1.2 A SUMMARY OF APPROACHES AND RESULTS

1.2.1 The Problem: Generating Writing Feedback

There are two main types of writing feedback from language tutors. General feedback comment on the more abstract aspect of the writing (e.g., is it fluent?). Localized feedback offer more concrete suggestions on improving a specific piece of writing (e.g. instead of "to," the preposition "for" should have been used in "a gift __ you".). My thesis focuses on modeling localized feedback.

Localized feedback may come from computers or humans. Computer programs are cheap at operation time; they are available 24/7 and fast to respond. However, current computerized systems can only handle a limited subset of grammatical errors, and with limited accuracy. Most of the current systems focus on single word confusion errors such as preposition errors (Tetreault et al., 2010a; Han et al., 2010; Dahlmeier and Ng, 2011b), determiner errors (Rozovskaya and Roth, 2010c) and verb mis-usage errors (Liu et al., 2010). For more comprehensive coverage, ESL learners still rely on language tutors.

While human tutors are much more capable, they need to stay focused on handling writing mistakes, which may take on many forms. One particularly challenging writing mistake to spot is *redundancy detection*. Unlike other frequent types of mistake (e.g. preposition/determiner mistakes), more than a single word might be redundant at a time (e.g., $\mathbf{Ex_1}$), and redundant words are not limited to a closed class set (e.g. verbs $\mathbf{Ex_2}$, nouns $\mathbf{Ex_3}$).

$\mathbf{Ex_1}$: There should be a careful consideration about what ~~are the things that~~ governments should pay for .
$\mathbf{Ex_2}$: . . . the price of crude oil will ~~keep~~ continue to go higher .
$\mathbf{Ex_3}$: So the ~~usage~~ amount of chemical substances will be largely reduced .

As a result, tutors cannot rely on obvious signals to indicate error prone areas; they must be constantly alert in order to identify these problems.

Just finding errors in ESL learners' writings is not enough – language tutors need to clearly indicate the location and reason for the writing mistakes. For example, in the sentence "He pick on a book", instead of suggesting that "pick on" needs to be replaced into "picks up", it would be better to indicate that two individual mistakes have occurred: a verb form error and a preposition mis-usage.

1. To fix the verb form error, we should replace "pick" into "picks".
2. To fix the preposition mis-usage, we should replace "on" with "up".

However, this is tedious in practice. Moreover, for ESL learners who are paired with a language partner instead of a language tutor, the partner may not even have the pedagogical vocabulary to describe the mistakes. Language partners prefer to edit the sentences in-place instead (Fregeau, 1999). For example, on popular language exchange social networking forums such as `lang-8.com`, language partners' feedback often do not indicate the location and reasons of the writing mistakes. Educational researches suggest that not telling learners enough information often demotivates them, harming their learning (Williams, 2003). Therefore, providing sufficient correction rationales is important for language learners.

Even building automated feedback generation systems requires repetitive efforts on the part of the language tutors. Many current automated feedback generation systems employ a supervised training framework. The idea is to "train" the system to act in a similar way as language tutors do, by tuning them with data containing language tutors' annotations. Language tutors need to annotate mistakes in many cases. For example, they need to mark that "for" needs to be replaced with "to" in the contexts "solution *for*", "give securities *for*", "harmful *for* people". Building certain system components however, does not require such annotations. I consider *confusion set*, an important component in Grammatical Error Correction (GEC) systems. Confusion set is a data structure that informs us which words are more likely confused with which. A confusion set is a rough figure of the instances of confusion. However, building confusion set often means that we have to collect statistics from a large annotated corpus.

1.2.2 My Approach: Computational Models of Tutor Feedback

This work's objective is to build computational models to reduce language tutors' efforts. In a nutshell, computational models are mathematical equations, and are designed to assign scores, classify and simulate. For example, underlying search engines such as Google and Yahoo!, a computational model helps choose which page is more likely to be the one we are looking for, by assigning scores to them (Page et al., 1999). In an automated essay grader, the computational model can assign a score to an essay by classifying it into different categories (Attali and Burstein,

2006). In a weather forecast system, a computational model simulates the movement of weather objects (e.g. clouds, wind), to help make its predictions (Lynch, 2008). When repetitive tasks can be described by a series of regular, predictable patterns, we may be able to develop computational models to automate them.

The aforementioned language tutors' tasks all have repetitive elements that computational models can handle. For redundancy detection, the computational model is a scorer that is applied repeatedly over every word; we can then select the word/phrase with the highest score as redundant (Chapter 4). For providing correction rationales, the computational model is a classifier that is applied repeatedly over each correction (Chapter 5). When building a GEC system, one repetitive work is to specify which words are most likely confused with a given word – building the *confusion sets*. In this case, the computation model is a simulation model that iterates over each confusable word (Chapter 6).

1.2.2.1 Models for Redundancy Detection

Redundancies are very common in ESL writings. Among the different types of feedback from language tutors in a learner's corpus, called NUCLE (Dahlmeier and Ng, 2011b), approximately 13% suggest addressing redundancy issues. By redundancies, I mean those words or phrases that do not add further meaning to a sentence, but instead make the sentence harder to read. Many redundant phrases can simply be removed directly without further changes.[1] I hypothesize that many of the redundant errors can be detected by a system that highlights the most redundant word in a sentence. Such a system might make a good first-pass filter to streamline a tutor's mental load on deciding whether to remove it.

To the best of my knowledge, there has not been any previous work that explicitly focuses on automatic redundancy detection. This may partly be because redundancy is a stylistic concept – I find that people tend to have their own opinions on whether certain words are considered redundant. To begin this work on more clear-cut cases, I have developed a corpus containing sentences in which the most redundant word is agreed upon by multiple annotators.

I have developed a system for this proposed task. At the core of my system is a model that assigns high scores to words that are more likely to be redundant. The scorer is trained using supervised machine learning techniques. In my exploration, I have identified two features to help

[1] Of the redundant phrases identified in NUCLE, 97% are suggested to be directly deleted.

describing how much a word contributes to the sentence – in terms of fluency and meaning. The fluency-centric feature is based on n-gram language models while the meaning-centric feature analyzes alignments between words and their translations into some other language. Selecting the word with the highest redundancy score is shown to have a 37% accuracy for identifying the most redundant word within a sentence known to have one.

Determining whether one sentence contains redundancies at all, however, is a different problem. We hypothesize the task needs a different feature set. I began the research on automating this task by first collecting clear-cut "good" sentences. Among the features we tried, sentence length and unigrams are the most predictive, together achieving a 70% accuracy in our balanced dataset.

1.2.2.2 Models for Correction Rationale The more tedious aspect of providing localized feedback is in the detailed markings of the location of the errors and citing the reasons for the changes. Building a system to automate interpreting these correction rationales is challenging because there may be several possible interpretations for one small change. For example, suppose a tutor rewrites the phrase "previous work" into "studies." Some possible rationales for this change include: (1) a single correction fixing a phrase mis-usage; (2) deletion of a redundant word ("previous"), followed by a word replacement ("work" $\Rightarrow$ "studies"); or (3) a word replacement ("previous" $\Rightarrow$ "studies") followed by the deletion of a redundant word ("work"). Although it is easy for humans to determine that the second interpretation is more plausible, this is less obvious to computers.

I hypothesize that there are common patterns in interpreting the rationales for tutor rewrites. We may build computer programs to automate these interpretations by memorizing these patterns. Such a system may auto-complete tutors' rewrites, allowing them to focus on coming up with the best rewrites.

I have built a model to help computers come up with the right set of corrections for 80% of the input sentences. The model focuses on determining whether two adjacent edits are fixing the same issue. Internally, it matches the adjacent edits against patterns such like "the edit is within the same word class" and "the edit is deleting a preposition". It then weighs the matching patterns, to make the right decision.

1.2.2.3 Models for Confusion Set Construction

Previously, most feedback generation systems have been developed using supervised machine learning methods. This often requires at least some parts of the system to work through "training examples," which must be annotated by a knowledgeable human, such as language tutors.

One component that is used in many grammar error correction systems is a module for constructing confusion sets. The confusion set of a word is the set of other words that students often confuse with the given word. The confusion set is needed so that a feedback generation system can select the most compatible word within the confusion set as an alternative. I hypothesize that confusion sets can be generated without the need for human intervention. Most of the confusions occur while learning the new language. Developing a simulation model for learning the new language would help capture which words are most commonly confused with each other.

I have employed a model that simulates the way words become confusable during learning. My model encodes words into vectors. If two words often occur within similar contexts, they are more likely to be confused with each other. I have adopted algorithms to adjust word vectors, to push together words that often occur within similar contexts. My empirical studies show that the resulting word vectors reflect confusions about word usage.

1.2.3 Contributions

1. I have shown the feasibility of detecting writing redundancy with computational models.
    a. I have constructed a balanced corpus to facilitate research on redundancy detection. This corpus contains "clear-cut" cases: half the sentences are judged by multiple annotators as containing no redundant words; the other half contain a word that most annotators agreed to be the redundant one.
    b. I have derived a measure for estimating the likelihood that a word might be redundant within a sentence.
    c. I have trained classifiers to detect redundant words in sentences. Through empirical validations, I have shown my proposed measure to contribute to the detection of the most redundant word within a sentence known to contain one.
    d. I have conducted experiments to investigate the interaction between redundancy prediction

at the word-level (how likely is a word to be redundant?) and at the sentence-level (does this sentence contain any redundant word?). My empirical results suggest that while the two are correlated, sentence-level redundancy predictions are equally correlated with other factors, such as the length of the sentence.

2. I have improved a computational model for detecting individual corrections within sentence revisions.
3. I have reduced the reliance on human annotations when developing a computational model for building *confusion sets*, an important component within a GEC system.

2.0 BACKGROUND

2.1 NATURAL LANGUAGE PROCESSING IN COMPUTER ASSISTED GRAMMAR LEARNING

Building computer systems that serve as automated tutors to help students to learn a new language has long been the focus of many research work in the area of Computer Assisted Language Learning (CALL) (Beatty, 2003). There has been a great deal of work in CALL aiming to help learners to improve in their listening, speaking, reading and writing skills (Levy, 2009). The most relevant area to this dissertation is on intelligent tutoring for writing; of particular interest are systems that help ESL learners with addressing grammar errors.

A key component within an intelligent tutor for writing is a mechanism to automatically assess student writing, detect any errors within the writing, and possibly propose corrections for these errors. To do so, many systems rely on natural language processing (NLP) techniques (Heift and Schulze, 2007; Leacock et al., 2010; Attali and Burstein, 2006). In the following sections, I review some recent NLP work in the area of grammar error detection and correction. I will first present the main types of problems that NLP methods try to address; then I will introduce some common computational methods used to tackle these problems.

2.2 GRAMMAR ERROR CORRECTION (GEC)

For a computer system to detect grammar errors, must it "understand the English grammar?" This is not an easy problem. One difficulty lies with defining what "English grammar" means. On the other hand, it is well known that learners are more likely to make certain types of errors (Leacock

et al., 2010). Among the most frequent errors are in the following four categories:

Preposition errors: . . . very promising and will likely $^{\text{to}}$ be the . . .

Determiner errors: However, as ~~the~~$^{\text{a}}$ result of $^{\text{the}}$ higher operating temperature of . . .

Verb choice errors: . . . more people are ~~acquired~~$^{\text{equipped}}$ with the knowledge . . .

Interleaving errors: . . . curb the problem ~~a population's aging and declining~~ $^{\text{of an aging and declining population}}$.

Because these errors have some regular patterns to them, it is possible for computers to automatically point out some of these errors to ESL learners. Instead of trying to teach computer what "English" is or is not generally, many grammar error correction (GEC) systems take an empirical approach and focus on specific grammatical challenges(Tetreault et al., 2010a; Liu et al., 2010; Rozovskaya and Roth, 2010c; Dahlmeier and Ng, 2011b).

2.2.1 Single-word Error Correction

Preposition and determiner errors are among the most frequent errors in ESL writings (Leacock et al., 2010; Dahlmeier et al., 2013; Yannakoudakis et al., 2011). One commonality between them is that these errors involve single-word choices between a small set of words. For example, the determiner mis-usage "as ~~the~~$^{\text{a}}$ result" is caused by the speaker mistakenly choosing "the" instead of "a". The confusion is more likely to happen between the determiners; but not likely between a determiner and another word, say between "the" and "theoretical". The mistaken insertions (e.g. ~~to~~ here) or deletions (e.g. will likely $^{\text{to}}$ be) can also be seen as confusions between an empty word ϵ and a preposition/determiner. Therefore, one way to solve preposition/determiner error correction is with a word selector (Tetreault et al., 2010a; Han et al., 2010; Gamon et al., 2008). The word selector chooses the right word from a *confusion set*. For example, *to*'s confusion set would contain *toward* and *for*. A common GEC pipeline is illustrated in Figure 1.

2.2.2 Phrasal Error Correction

Not all errors occur independently on single word level, or between small word sets. In the *Verb choice errors* example, the confusion is between verbs in a open class. In the *interleaving errors* example above, we cannot attribute the error to one single word choice. For *interleaving errors*,

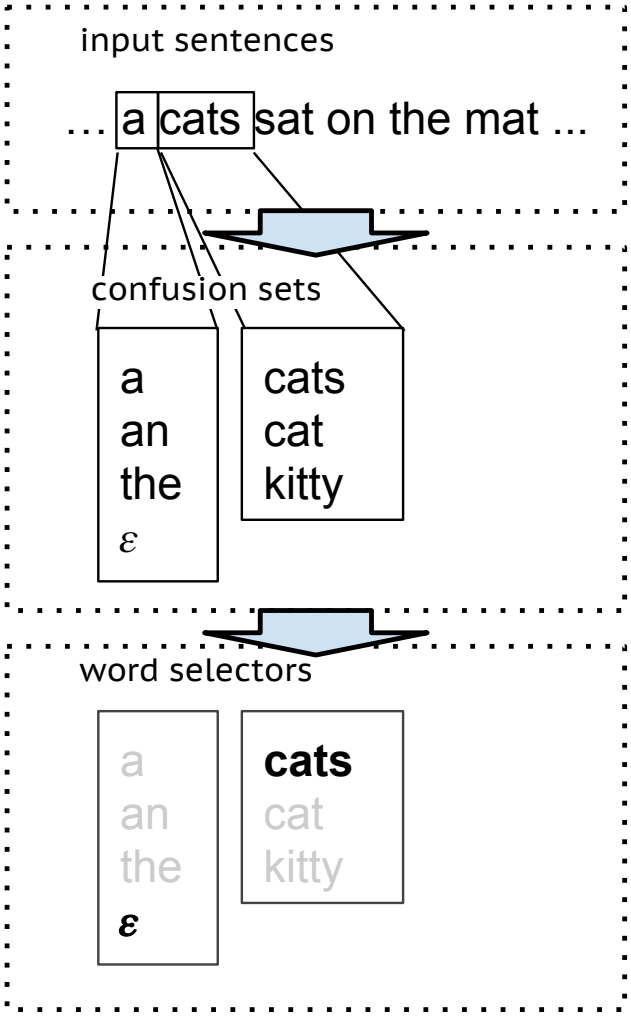


Figure 1: For single word mistakes, the GEC system considers each word individually – first looking up that word's confusion set, then using the word selector to choose the most appropriate replacement.

although we may decompose a correction into multiple single word choices, these single word choices are dependent on each other. For example, the following correction

~~a population's aging and declining~~ of an aging and declining population

can be decomposed into: (1) inserting an *of* (2) replacing *a* into *an* (3) deleting *population's* (4) inserting *population*. These fixes are not independent: (1) replacing *a* into *an* has to happen once we decide to delete *population's* (2) deleting *population's* need to co-occur with inserting *population* in the end. For open class errors, the word selector would need to select from many candidates, making its task even harder.

To correct these errors, recent work have employed two strategies. The first strategy still focuses on correcting individual errors, but additional models are used to detecting the dependencies between errors. This can be done with sequence models such as Hidden Markov Models (HMMs) (Gamon, 2011) and linear programming (Wu and Ng, 2013). A second strategy is to employ more flexible models that can rewrite a phrase/sentence entirely. Recent researches applied machine translation models into solving these problems (Knight and Chander, 1994; Brockett et al., 2006; Xue and Hwa, 2010; Madnani et al., 2012b). In this paradigm, the error correction problem is formalized into "translating from a non-native language into well-formed English".

2.2.3 Correction Detection

A related GEC task turns the detect-then-correct pipeline on its head; instead, it tries to predict the rationale behind a particular correction. If a system were given the *interleaving errors* example from above, its output ought to be: (1) *of* is added to fix the missing preposition mistake (2) *a* needs to be removed for violating determiner usage rules. (3) *population's aging and declining* needs to be reversed to make *population* the center noun. We may use computer programs to automate this – we call this task *Correction Detection* (Swanson and Yamangil, 2012; Xue and Hwa, 2014). In *correction detection*, the computer first figures out how many individual errors are fixed. Then the computer will then categorize these individual fixes based on their reasons, with a *correction type categorizer*.

Shared task	error types	corpus
HOO 2011	All "light copy-edits", including spelling errors, lexical choices, basic grammar errors, reduction of syntactic complexity etc.	HOO 2011
HOO 2012	preposition and determiner errors	FCE
CoNLL 2013	preposition, determiner, noun number, verb form, subject-verb agreement errors	NUCLE
CoNLL 2014	All error types	NUCLE

Table 1: Error types and corpora being used in recent HOO and CoNLL shared tasks.

2.2.4 Shared Tasks

Recently there have been competitions among GEC researchers, which helped boosting discussions within the community. These competitions are built around shared tasks, where organizers build a common playground to evaluate participating systems' performance. Two series of shared tasks are HOO (2011 and 2012) and CoNLL (2013 and 2014). The tasks include building GEC systems for all types of grammar errors (Dale and Kilgarriff, 2011; Ng et al., 2014), or a common subset (Dale et al., 2012; Ng et al., 2013). The task organizer put up corpora to evaluate participating system's output. Parts of these corpora are also distributed to the participating teams to help them develop GEC systems. I summarize the error types and corpora being used in these shared tasks in Table 1.

These shared tasks introduce evaluation metrics to compare GEC systems from different angles. A common framework is to compare system's output with a gold-standard, using F_α scores:

$$
\begin{aligned}
F_\alpha &= \frac{(1+\alpha^2)PR}{\alpha^2 P + R} \\
P &= \frac{\text{number of system's output matching with the gold standard}}{\text{size of system output}} \\
R &= \frac{\text{number of system's output matching with the gold standard}}{\text{size of the gold standard}}
\end{aligned}
$$

By default we use $\alpha = 1$ to achieve a balance between precision (P) and recall (R); but recently people lean toward reducing systems' false alarms (e.g. CoNLL 2014), and therefore prefer lower α values (e.g. 0.5). Depending on whether the goal is to let GEC systems detect (i.e. find error locations), recognize (i.e. categorize the error) or correct writing mistakes (i.e. fix the error), we may define "matching" differently (Dale and Kilgarriff, 2011; Dahlmeier and Ng, 2012). To provide a more comprehensive view of different systems, organizers often report evaluation results on multiple aspects. For example HOO 2011 evaluated participants' systems on three perspectives: detection, recognition and correction.

So far the winning teams have been using classification based models as well as machine translation based models. The winning teams in HOO 2011 (Rozovskaya et al., 2011), HOO 2012 (Dahlmeier et al., 2012) and CoNLL 2013 (Rozovskaya et al., 2013) all built upon classifiers trained to correct single word errors. The common features in their systems include n-grams, POS tags and phrase chunks. Please note that even the HOO 2011 task includes all error types, the leading team did not try to handle them all at once. They instead build classifiers for the most common error types including preposition and determiner errors. Recently there is a trend to apply translation based models for broader error types. The winning team in CoNLL 2014 (Felice et al., 2014), for example, applied a hybrid system containing manually crafted rules, machine translation systems and language models.

2.3 COMPUTATIONAL MODELS FOR GEC PROBLEMS

Computational models are mathematical equations that, when taken together, formalize how we make decisions in certain contexts. In GEC, for example, we need them to decide (1) whether to report an error, and (2) how to correct it. Computational models often contain many parameters, which are tuned by "training" to optimize the model's performance for a particular task. Training is often performed on datasets, containing example decisions. GEC systems are trained on datasets that contain real ESL writings, along with language tutors' annotations, including the locations of errors and how to correct them. Because datasets are limited, researchers also seek for ways to reduce the number of model parameters.

2.3.1 Classification Models

A classifier assigns something into one of several pre-defined categories by considering its *features*. For GEC tasks, classifiers can be used to implement a word chooser or an error type categorizer. For example, in the phrase *as* **the?** *result,* a determiner-chooser has to select an appropriate word from a list of possible determiners for the position currently taken up by the word **the?**; two possible features to help it to make the decision might be: the word before the position (in this example: *as*), and the word after it (in this example: *result*).

Computationally, the classification problem is typically formalized as maximizing the likelihood that the categorizations for a set of training examples are correct (Bishop et al., 2006). Formally, suppose we want to choose the right determiner w under some context c, the classification decision is to find the most appropriate word:

$$\bar{w} = \arg\max_{w \in \{\text{the, this, an, } \ldots\}} \Pr(w|c, \lambda)$$

Here λ is the classifier's parameters. Different classifiers have their own definitions of $\Pr(w|c, \lambda)$. For example, in a Maximum Entropy model, the classification probability is expressed as:

$$\Pr(w|c, \lambda) = \frac{\exp \sum_i \lambda_i f_i(w, c)}{\sum_{w'} \exp \sum_i \lambda_i f_i(w', c)}$$

Here each feature function f_i extracts a value $f_i(w, c)$ for a determiner choice w under context c. For example, we may define a feature f_1 to determine whether the previous word, current word and the next word will together form the phrase "as a result".

2.3.2 Sequence Labeling Models

Hidden Markov Model (HMM) and Integer Linear Programming (ILP) are two ways to express dependencies between individual decisions. HMM can be used to detect error regions for GEC problems(Gamon, 2011). ILP can be used to make sure the system proposes corrections that do not conflict with each other (Wu and Ng, 2013). When detecting error regions, the problem becomes making decisions on whether each word is in the error region or not. On top of knowing which words are more error-prone, HMM helps us incorporate knowledge such as (1) a determiner error is likely not immediately followed by another determiner error (2) it is rare to have too many

corrections in one region. ILP is similar with HMM, with the slight difference that it enforces hard constraints. As a result, an HMM "discourages" two consecutive determiner corrections, but ILP "prohibits" two consecutive determiner corrections.

Computationally, ILP represents decisions (e.g. whether to replace a "the" into "a") as binary variables ("yes" or "no"), and formalize relations between these variables. Take (Wu and Ng, 2013)'s treatment on "*a cats* sat on the mat" as an example. We may represent whether we want to fix "*a*" as a determiner mistake as one variable $x_1 \in \{0, 1\}$; at the same time, we may represent whether we want to fix "*cats*" as a plural-form mistake as another variable $x_2 \in \{0, 1\}$. In ILP, we may enforce that $x_1 + x_2 \leq 1$ to make sure that we do not apply these two fixes at once. This helps us to deal with sentences that have interleaving errors .

HMM enables us to formalize the relation between two adjacent labels. Gamon (2011) uses a sequential labeling model to detect error regions. Here they annotate the error parts as "I", and correct parts as "O". Error detection then becomes a sequence labeling problem. During sequence labeling, our algorithm will search for the sequence of labels has the highest probability. The probability is proportional to the product of (1) emission probability and (2) transition probability. Emission probability is the probability of, for example, the word "of" is in the error region. Transition probability is the probability of, for example, one word labeled "I" followed by another word labeled "I". HMM has variations that enable it to incorporate more features. These variations include Conditional Random Fields (CRF) (Lafferty et al., 2001) and Maximum Entropy Markov Model (MEMM) (McCallum et al., 2000) (which is applied in Gamon (2011)).

2.3.3 Machine Translation Models

Machine Translation (MT) concerns translating texts within one language into another. It has received much attention not only because of the importance of translation itself, but also because of its ability to generalize to other tasks (Wubben et al., 2012; Coster and Kauchak, 2011). In GEC systems, for example, we treat the "ill-formed" language by ESL learners as a special language. Correcting grammar mistakes then becomes "translating" from ill-formed English into well-formed English (Knight and Chander, 1994; Brockett et al., 2006). Because machine translation concerns full sentence rewriting, these resulting systems can be made capable of handling

complex error types. The training corpora can be either collected from human annotations, or from automatically generated corpora, where we populate errors into well written texts (Foster and Andersen, 2009; Brockett et al., 2006).

Many machine translation models are also probabilistic models, which assign a probability to a pair of sentences (Brown et al., 1993; Koehn et al., 2003). The system will translate by finding the sentence with the greatest translation probability. For example, consider translating a Foreign language sentence f into English. A translation model can calculate the probability $\Pr(e|f)$, that e is a translation of f, for any English sentence e. During translation, a machine translation system searches for $e' = \arg\max_e \Pr(e|f)$.

One straightforward way of constructing the translation model is to build a huge table containing translation probabilities between any two sentences in the two languages. But this is impractical because of the near-limitless number of potential sentences in both languages. We often seek to decompose $\Pr(e|f)$ in ways that it be computed from smaller components. In this thesis I in particular consider word-alignment models (Brown et al., 1993) and phrase-based translation models (Koehn et al., 2003), two most frequently used models that formalize this idea. Instead of building a table of sentence-level translations, word-alignment build a table of how words in two languages map to each other. Phrase-based translation models is even more general in that it builds a table of corresponding phrases in two languages. During operation, word-alignment based systems looks up each word's translation in the translation table, and groups them into a fluent sentence in the target language. Phrase-based MT systems operate similar except that they need to chunk the input sentences before phrase look-ups. Google translate [1] adopts a phrase-based model. Also, we can use the open-source tools GIZA[2]++ and Moses [3] to build customized MT systems (Koehn et al., 2007).

2.3.4 Multi-task Learning and Domain Adaptation

One challenge in building GEC systems is that we have many model parameters, but limited training data. As a result, the trained models may over-fit. One way to relieve this problem is to reduce

[1] http://translate.google.com/
[2] https://code.google.com/p/giza-pp/
[3] http://www.statmt.org/moses/

the number of model parameters. The key observation in GEC systems is that many tasks are similar: either between GEC tasks, or between GEC and other tasks. For example, correcting the misused *of* is similar with correcting the misused *to*, in that the right choice under the context might be the same. Also, GEC is similar with machine translation in that they both involve rewriting a sentence into a grammatical target sentence.

Multi-task learning allows us to share parameters between different GEC tasks. Dahlmeier and Ng (2011b) uses Alternating Structure Optimization (ASO) (Ando, 2006) that formalizes the relation between preposition error correctors. Normally we would have 36 word selectors for detecting the mis-usage of every frequent preposition (e.g. the mis-usage of *to* etc.). We would therefore need 36 separate sets of parameters. Alternating Structure Optimization compresses these parameter sets into one shared set, using Singular Value Decomposition (SVD). It then augments the shared set with a smaller set of parameter for each individual preposition mis-usage corrector. In the end, the total number of free variables will be reduced.

Domain Adaptation allows us to build GEC models on top of models for similar tasks. In domain adaptation, instead of starting from scratch, we only model the difference between the new tasks and the existing task. Two models that GEC researches commonly build upon include language models and paraphrasing models. Language models can tell us the probabilities of a word (e.g. *a*) occurring under a certain context (e.g. *as __ result*) in native English. Paraphrasing models can rephrase the input sentence into a set of other English sentences that have the same meaning. To adapt these models to GEC systems, we come up with models for picking the output from these models that are likely to fix grammar errors. To adapt a language model for GEC purposes, we model what errors are more likely to occur in ESL writings. In Park and Levy (2011), the authors uses the error model and the language model to together choose corrections that: (1) proposes to fix an error that is likely to occur (2) proposes a correction that matches English grammars. To adapt a paraphrasing model for GEC purposes, we may use models to combine the paraphrase that best fixes the problems . In (Madnani et al., 2012b), the authors in particular paraphrased the input sentence by round-trip translation: first translating the input sentence into another language, and then back to English. Then, the authors combined the paraphrases via multiple languages by first aligning them together, and then developing metrics to choose the best combination. Because the underlying language models, translation models are well-studied, the combined model tend to

work well even with a rough domain adaptation model.

2.4 GEC CHALLENGES

Relying on a fully automated system to provide writing feedback to learners is still impractical as of now. Current systems are limited in both coverage and accuracy. Many systems focus on one specific error type rather than interleaving error types (Chodorow et al., 2007; Tetreault and Chodorow, 2008; Gamon et al., 2009; Liu et al., 2010; Rozovskaya and Roth, 2010d; Dahlmeier and Ng, 2011b). Moreover, even within preposition error detection, systems do not perform with a high accuracy. In a recent shared task, HOO2012 (Help Our Own), the best performing system is reported to have a 33.23% F_1 score (Dale et al., 2012). At this level of performance, a system cannot reliably correct all learner errors; therefore, language students still need to consult human tutors.

To move GEC systems forward, several challenges needs to be addressed. First, systems ought to cover a wider variety of error types. This may mean that we need to consider different computational models. Despite a wide variety of computational models we already have (Section 2.3), adapting them to new error types often need much research/development effort. Second, there ought to be better data to support building GEC systems. Data on real ESL errors help us to both develop and evaluate GEC system building. As I will discuss in Chapter 3, current datasets are limited in both their qualities and quantities. Finally, in addition to developing fully automated systems, we also ought to come up with better ways to support language tutors to perform their tasks.

3.0 DATASETS AND SOFTWARES

As discussed in the previous chapter, the success of many NLP methods depend on having the appropriate data resources. In this chapter, I present an overview of the existing datasets that are relevant to the development of GEC technologies. I discuss the extent to which these resources could be used to support the work in this dissertation. I also present some common software packages used in this thesis work.

3.1 GEC DATASETS

Data-driven approaches to GEC require realistic examples of student errors. Therefore, a particularly useful type of dataset is a *learner's corpus*, a collection of learners' texts that has been annotated with suggested corrections. Figure 2 illustrates a portion of the learner's corpus, the NUCLE corpus (Dahlmeier et al., 2013). In learners' corpora, teachers will mark errors in learners' writings, suggest fixes for them. To help learners understand the rationale, they also annotate the error type that has been fixed.

3.1.1 Desiderata

At the core of many modern GEC systems is a supervised machine learning method. The annotated examples serve as training instances for the system to learn to make predictions so that it can perform language tutors' tasks. A good corpus, therefore, should help us to train computers to generate corrections as human tutors do, and to validate our systems by comparing the systems' outputs against the human annotated examples.

As technology continues to advance at an ever increasing rate, energy is also being consumed at a rapid pace. Oil is still the main source of energy that drives the economy despite its ill effects on the environment. Furthermore, it is projected to be depleted in fifty ~~years~~years'(Npos) time. Therefore, the global issue of today has to do with clean and sustainable energy. The six Generation IV nuclear reactors that are currently being researched under the supervision of the Generation IV International Forum (GIF) are the answers to this very problem. As of now, the Lead-Cooled Fast Reactor (LFR) and the Sodium-Cooled Fast Reactor (SFR) ~~are looking~~appear(Wtone) very promising and will likely ~~to~~(Vform) be the main choice of nuclear power plant in the near future. However, further comparison of both the reactors ~~base~~based(Vform) on cost, performance and safety actually revealed (LFR) as the better option.

Figure 2: Example annotations in NUCLE corpus. English teachers have marked error words/phrases (crossed-out red words), and their corrections (green words). Along with each correction, teachers will also specify the error type. Take the first correction as an example: the teacher suggested to revise *years* into *years'* to fix a noun possessive form error.

There are many questions to address with respect to the exact form of the annotations, however. Consider the following sentence:

> However, have you ever thought that: what if it poses a great challenge for the public?

The three underlined parts read awkward. The phrase "have you ever thought that:" reads wordy, should we keep it, or delete it, or rewrite the sentence into "However, it may pose ..."? The preposition "for" needs to be replaced by "to", should we tell the learner why? If so, should we say it is because "for" is misused preposition, or should we say it collocates incorrectly with "challenge"? The phrase "the public" is too vague: if we decide to also replace into "public resources", should that be treated as one whole correction, or two separate corrections (i.e. deleting "the" and then adding "resources")?

In general: (1) What types of errors should be corrected? Should we only consider grammatical errors, or also awkward usages? Note that whether something is awkward may be subjective; there may also be multiple ways of correcting it. (2) Should the annotation be just the corrected phrase or should it also include a rationale for the correction. (3) If a category is given for each corrections, how finely should different error types be distinguished? (4) How should complicated interleaving errors be addressed?

More broadly, in terms of creating a learner's corpus, there are additional decisions to be made. First, who should do the annotation? Here are some options:

1. Language tutors: They provide high quality annotations (Dahlmeier and Ng, 2011b); the quality can be further improved when the same text is annotated by multiple language tutors (Nicholls, 2003). However, this is the most expensive option. Therefore, the resulting corpus may not be very large.
2. Language peers. An example of this is a social network learning forum like Lang-8.com (Mizumoto et al., 2011). These social network system have large user bases, and so is the volume of annotations. However, it is harder to control the quality of the annotations, as the backgrounds of both learners and annotators often vary.
3. Crowd-sourced non-experts from the Internet (e.g., Amazon Mechanical Turkers). This way of gather annotations is quick and cheap, but the overall quality may not be as reliable (Tetreault et al., 2010b).

Second, what are the characteristics of the collection of learner texts themselves?

1. the learners' native language (L1). This is an important factor because error types/frequencies are influenced by learners' L1's (Rozovskaya and Roth, 2010c).
2. the learner's level of proficiency. A collection of texts from highly proficient learners will have different error distributions than a collection of texts from learners who just started studying the new language.
3. genre of the text. The distribution of the types of errors may also be influenced by Whether the text is a general composition or an argumentative essay or a technical articles.

For each of the above issues, one may opt to develop a balanced corpus (e.g., an equal representation of learners with different L1s) or not (e.g., only include texts by highly proficient learners).

3.1.2 Available Corpora

In Table 2, I enumerate some widely available corpora and compare their corpus design decisions. Depending on the target application we plan to build, using these corpora expose different advantages. When building GEC systems on top of machine learning systems trained on large datasets,

Corpus	Error Categorization	Annotators	Learners
NUCLE (Dahlmeier and Ng, 2011b)	27 types	English Teachers	Singaporelean college students.
Lang-8 (Mizumoto et al., 2011)	No	Language Peers	ESL learners from various countries.
FCE (Yannakoudakis et al., 2011)	75 types	English Teachers	Students taking Cambridge English exams. This is a subset of Cambridge Learner Corpus, .
UIUC (Rozovskaya and Roth, 2010b)	8 types	English Experts	Students from various language backgrounds. This is a subset of ICLE (Granger et al., 2002) and CLEC (Gui and Yang, 2003)corpora.
HOO2011 (Dale and Kilgarriff, 2010)	38 types	Scientific Article Editors	ESL scientific writers. This is from ACL Anthology Reference Corpus.

Table 2: A summary of annotated ESL corpora that are used in this thesis. NUCLE corpus is large and contains error categorizations, so it is a very good resource for tuning systems that target at specific error types. Lang-8 contains a huge number of revisions from peers online, making it a good resource for tuning more flexible/general GEC models. FCE contains many grammar error annotations, along with overall scores of each articles. This makes FCE also a good development corpus for research on the correlation between grammar errors and overall English sophistication. UIUC corpus contains high quality annotations on essays written by students of different background, making it a good resource for researches on native language (L1)'s influence on grammar errors. Unlike other corpora, HOO2011 actually contains articles written by highly sophisticated writers – scientific article writers. This makes HOO2011 a good resource for building systems for sophisticated ESL speakers.

Lang-8 and NUCLE can help, as they contain the biggest volume of annotations. When building GEC systems focusing on explaining the rationales behind each corrections, FCE can help, as it has carefully constructed fine-grained error categorizations. When evaluating GEC systems on detecting preposition/determiner errors, UIUC can help, as it contains high-quality annotations constructed by three sophisticated English experts. When building GEC systems for highly sophisticated users, HOO2011 can help, as it contains editors' suggestions on highly sophisticated articles.

Researchers have used GEC datasets in different scenarios, in addition to building GEC systems. Yannakoudakis et al. (2011) uses the FCE corpora for building automated scoring systems. Their model incorporates language learners' grammar error rate as a factor in predicting the learner's text score. Swanson and Yamangil (2012) uses these corpora to study how to generate suggestions and error categorizations from revisions. They use tutors' suggestions in annotated corpora as the gold standard.

3.1.3 GEC Datasets Used in This Thesis

In this thesis, most experiments are conducted on the NUCLE and FCE corpora because they are considerably larger than others, and because they contain real ESL learners' writings with English experts' annotations. However, existing annotations do not sufficiently or robustly identify redundancy errors (Chapter 4). In order to have an appropriate dataset to validate my experiments, I have commissioned additional annotations to be performed on NUCLE sentences so that unreliable cases may be filtered out. To address the problem of the lack of a unified coding standard, I use HOO2011 and UIUC in supplemental experiments for additional validations.

3.2 SOFTWARE PACKAGES USED IN THIS THESIS

I have conducted my thesis experiments with the help of several off-the-shelf software packages for some common NLP and machine learning tasks such as part-of-speech tagging, syntactic parsing, machine translation, and classification.

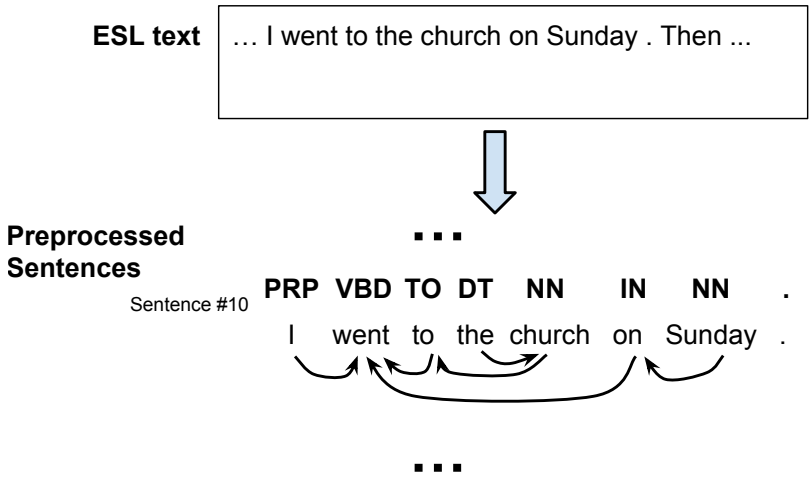


Figure 3: After preprocessing, we will have individual sentences, individual words, their POS tags, the dependency structure between words.

3.2.1 Preprocessing Software Packages

All the sentences in the datasets have been preprocessed with the OpenNLP toolkit[1] and the Stanford parser (Klein and Manning, 2003). The OpenNLP toolkit is used to perform some preliminary analysis of the text. For example, it breaks up a student essay into a collection of individual sentences; it also assigns a part-of-speech tag (such as *noun, verb, determiner, etc.*) for every word in each sentence. The Stanford parser is used to provide syntactic analyses of the sentences. Specifically, I used it to generate the dependency structure of each sentence. Figure 3 illustrates the my preprocessing pipeline.

3.2.2 Software Packages for Classification

Classification is an integral part of this dissertation. It is needed for tasks throughout the thesis, including:

1. binary classification of whether a word is likely to be redundant or not (Chapter 4);

[1] https://opennlp.apache.org/

2. binary classification of whether two adjacent edits address the same grammar error (Chapter 5);
3. multi-label classification of choosing the most appropriate preposition under a specific context (Chapter 6).

I choose to work with a probabilistic classifier – Maximum Entropy classifier (MaxEnt, or log-linear model) for these purposes; in addition to giving a classification, it also gives a likelihood estimate for its answer. The particular implementation I used is in C++/Python, and it is written by Le Zhang[2].

3.2.3 Machine Translation Packages

Translators are used in this thesis to gather additional features for our models. In particular:

1. In Chapter 4, I use translators to examine the number of words that an English word would translate into. This helps our model to roughly examine the word's importance.
2. In Chapter 6, I use translators to examine what a preposition would translate into. This helps our model to "understand" whether the preposition is misused.

There are several statistical machine translation systems that are publicly available. I chose to work with Google translate [3] because it provides decent translations along with the word alignments between source/target sentences. An example of its translation output is shown in Figure 4.

Additionally, post-processing on the translated foreign language is sometimes necessary. For instance, many Asian languages, such as Chinese and Japanese, do not use space delimiators to indicate word boundaries. For example, the Chinese sentence in Figure 4 would be displayed as: "我周日去得教堂。" In these cases, a word tokenizer is needed to split up the sentence into words, e.g. "我␣周日␣去␣得␣教堂␣。" I used a state-of-the art tokenizer Chang et al. (2008) to tokenize Chinese translations.

[2] http://homepages.inf.ed.ac.uk/lzhang10/maxent_toolkit.html
[3] http://translate.google.com

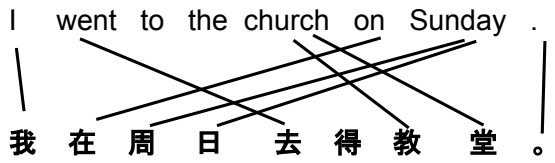


Figure 4: With Google translate, we are able to fetch sentences' translation, along with the alignment between the words in both languages. This figure in particular illustrates the alignment between English words and their translations in Simplified Chinese.

4.0 LOCAL REDUNDANCY DETECTION

4.1 CHAPTER OVERVIEW

"Writing concisely" is a time-honored advice for writing. However, ESL writings often contain unnecessary or redundant words and phrases. Because ESL learners are still inefficient at communicating in English, perhaps they use extra words to ensure that they communicate their messages; or perhaps they do not have the right language model to know when to refrain from using connective words. Sometimes, the extra words are semantically unnecessary; other times, they violate English grammar rules. Below are some examples of what we refer to as *local redundancies*.

Ex$_4$: ... the usage amount of chemical substances will be largely reduced ...
Ex$_5$: ... is a new term for us. ...
Ex$_6$: ... the citizens can request for higher salaries ...

In **Ex$_4$**, *usage* is implied when we mention *amount of chemical substances*. In **Ex$_5$**, *for us* is extraneous because readers would not assume otherwise. In **Ex$_6$**, *for* should be removed syntactically: "request" should go directly before its object; semantically, the meaning "for" is already implied by "request".

Analyzing existing ESL corpora, we observe that a significant proportion of the feedback from language tutors is on removing local redundancies. In the NUCLE corpus (Dahlmeier and Ng, 2011b), 13.7% of the annotations are tagged as "local redundancy errors" (Rloc), making it the second most common feedback type. In the FCE corpus, 14.0% of errors are tagged starting with "U" (unnecessary). To help reducing the efforts of language tutors, my objective is to develop a redundancy detector. Can a system automatically identify words or phrases that a tutor will likely eliminate?

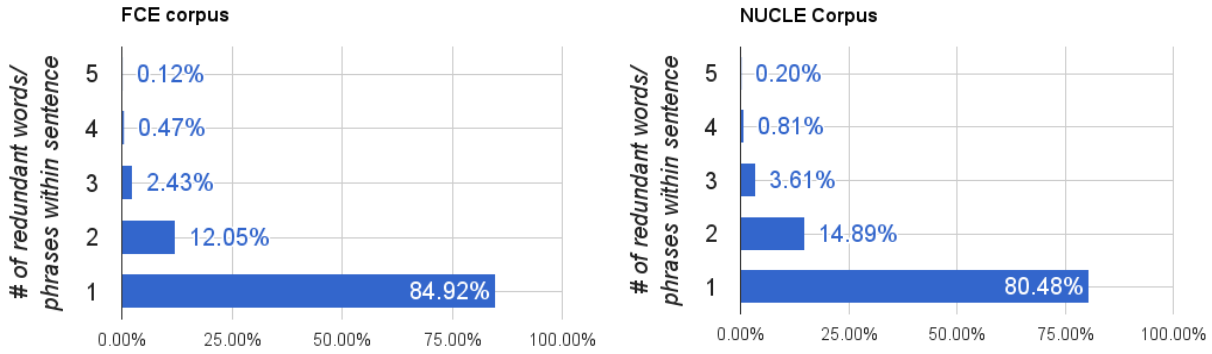
While previous work in the GEC literature (Tetreault et al., 2010a; Rozovskaya and Roth, 2010d) do handle some cases of redundancies in which grammatical rules were violated (e.g., $\mathbf{Ex_6}$), my work is the first to explicitly focus on redundancy detection. I investigate two questions: whether a system can automatically identify the most redundant word in a sentence; and whether a system can determine if the sentence contains any redundant word. A machine learning approach is taken in both cases. Experimental results show that the method I developed can identify the most redundant word within a sentence known to contain one 37% of the time, which is 7-times more accurate than a baseline of random choice. For the broader distinction of whether a sentence contains any redundant word, my classifier predicts correctly with 70% accuracy on a balanced dataset.

4.2 MAIN PROBLEM: SINGLE WORD REDUNDANCY DETECTION

Annotations from existing ESL corpora suggest that language tutors rarely identify more than one region within a sentence as redundant (Figure 5). Moreover, most redundant phrases consist of just a single word (Table 3). As a first step toward developing a fully automated redundancy detector, my work focuses on the prevalent cases of single occurrences of a redundant word within a sentence: A successful detector should highlight the most redundant word (if any) within an input sentence. Even within this more restricted problem space, redundancy detection is still very challenging.

As we have discussed earlier, a word might be considered "redundant" for many reasons. It is not restricted to any particular part-of-speech category (Figure 6 shows the distribution of part-of-speech categories of redundant words in two corpora); it might be removed for grammatical reasons, for semantic clarity, or for brevity. My work does not explicitly distinguish between different reasons for redundancy because I believe they all still share some common patterns. Namely, the removed words do not contribute as much as other words in the sentence toward the overall fluency or meaning of the sentence.

On the other hand, this work does not aim to address *all possible cases* of single-word redundancy either. As we have discussed earlier, judgments of some cases of redundancy are subjective.



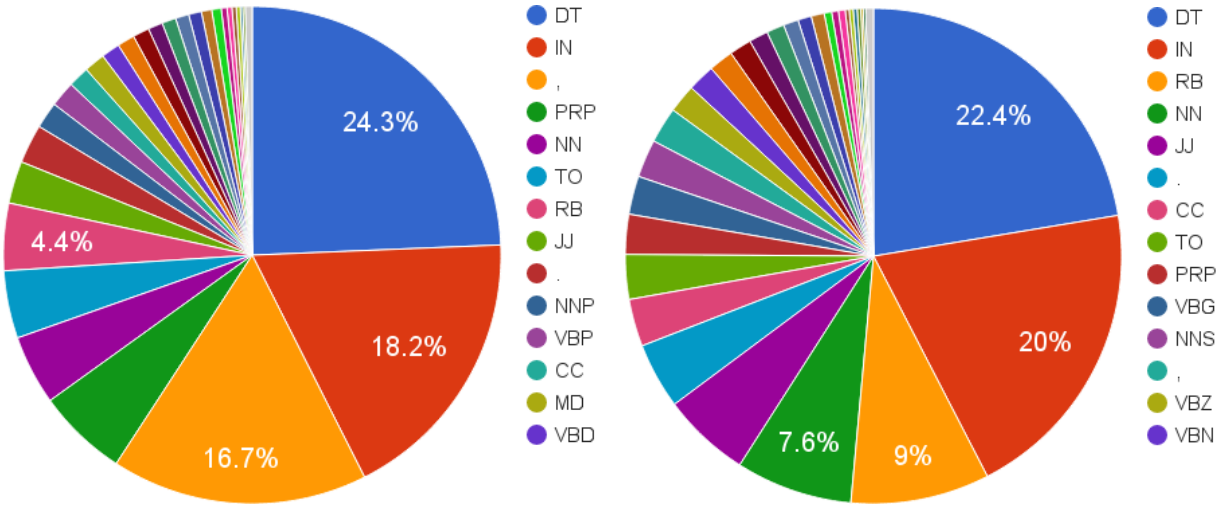
(a) Distribution bar chart in FCE corpus.

(b) Distribution bar chart in NUCLE corpus.

Figure 5: Distributions of the number of local redundancy feedback within single sentences, in NUCLE and FCE. This distribution graph only includes statistics from sentences with at least one redundant word/phrases. In both corpora, the majority of these sentences contain one redundant region.

Corpus	Percentage of single word redundancies
NUCLE	67.55%
FCE	86.93%

Table 3: Of all the phrases the language tutors marked as "redundant", the majority has a phrasal length of one.



(a) FCE corpus.

(b) NUCLE corpus.

Figure 6: Distribution of single word redundancies over word classes in annotated corpora. The minority word types add up to a large proportion.

Consider $\mathbf{Ex}_4$ again. When asked the following two questions about it, people do not have uniform responses.

1. Which word in this sentence do you think is the most redundant?
2. Does this sentence contain a redundant word?

For the first question, among five native English speakers I asked, three found *usage* more redundant while the other two found *amount* more redundant; still there may be people who pick some entirely different word in the sentence (e.g. *chemical* or *largely*). Moreover, even if two people agree on "the most redundant word" in a sentence, it does not mean that they agree on the severity of the redundancy within the sentence. One might acknowledge that some word's role in a sentence is not strictly necessary but still not want to delete it.

To make the problem more tractable, my research examines redundant cases that most people would agree on. Moreover, I tackle each of the above questions separately so as to minimize the influence of subjectivity from the other question. I refer to addressing the first question as a **word-level** sub-task; and the second as a **sentence-level** sub-task. There are many clear-cut cases in each

sub-task, where people tend to agree. For example, most people would agree that "also" is the most redundant word in "I like apples and also oranges" and that the sentence "I like apples" is not redundant. If both sub-tasks can be successfully accomplished, their solutions point to a system that forms a detection pipeline for identifying single word redundancies. My work focuses more on the first sub-task than the second because the development of a computational model that can reliably estimate the potential for redundancy of a single word might be considered a prerequisite for deciding whether a sentence contains any redundancy over all.

4.3 TASK 1: WORD-LEVEL REDUNDANCY DETECTION

4.3.1 System Framework

To identify the most redundant word within a sentence, we first measure each word's potential for redundancy, then pick the highest one. At the core of this system is a component that assigns a *redundancy score* to a word. We may use arbitrary measures (e.g. proportion of the word marked to be redundant in NUCLE) directly as the scorer. I also propose a supervised machine learning model to incorporate multiple features.

Figure 7 is an illustration of the proposed system's infrastructure. The scorer is developed using a probabilistic classification model: the Maximum Entropy (MaxEnt) model (Malouf, 2002). I train the model to predict the likelihood (between 0 and 1) that a word might be the most redundant one in its sentence. Like other machine learning endeavors, having appropriate training examples (details in Section 4.3.2) and features (details in Section 4.3.3) are the key to the success of the redundancy detector.

The training examples are from sentences in which one word has been manually annotated as the most redundant. As illustrated in Figure 8, for a sentence of n words, I construct n training instances, where each instance corresponds to one word. The redundant word corresponds to the only one instance where we want the MaxEnt model to output 1. Other words will turn into instances where we want the MaxEnt model to output 0.

The MaxEnt model makes its predictions using features extracted from the given word and its

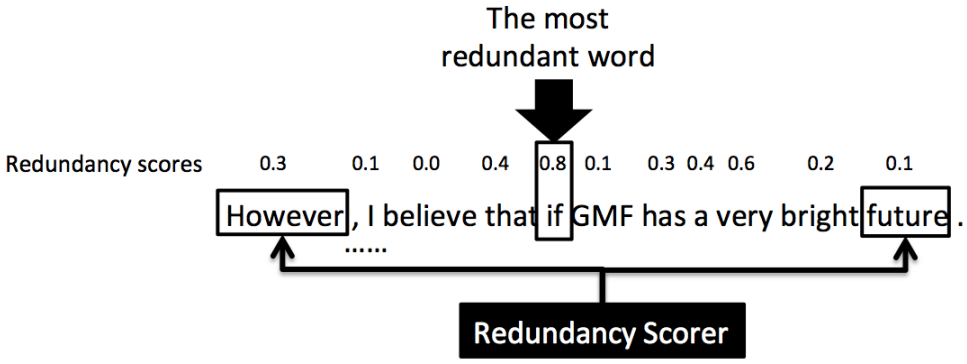


Figure 7: Redundant word detector system infrastructure. The system assigns a redundancy score to every word in the input sentence. The word with the highest score is picked as the most redundant word. The key component is the redundancy scorer.

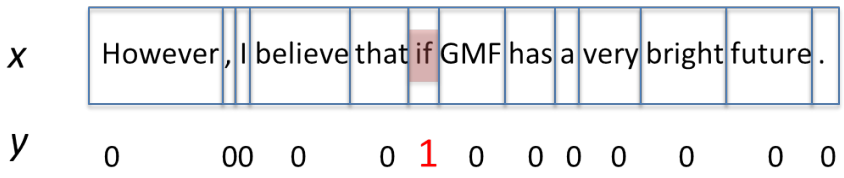


Figure 8: Constructing training instances from an annotated sentence.

context. Some example features are: a binary value indicating whether the given word occurs more than once in the input sentence; a numerical value representing the language model score of the sentence if the given word were omitted.

4.3.2 Corpus Development

Because current ESL corpora are not developed with redundancy detection in mind, the annotated instances within those corpora (e.g. "Rloc" in NUCLE, and "U" in FCE) are subject to inconsistencies and subjective judgments. As a result, the same word/phrase may be annotated differently in similar contexts. For example, consider the word "quite", which appears in the following three contexts, all following the same pattern: "a quite *adj. noun*":

. . . it is a ~~quite~~ safe design . . .
. . . a quite small mistake . . .
. . . to ~~a quite large~~ quite a large (adjective word order error) extend.

Though *quite* is used in similar ways in all three contexts, the language tutors' annotations are different. For the purpose of this study, these cases should be excluded.

To develop a more clear-cut corpus of redundant words and phrases, I decide to further filter the NUCLE corpus to obtain a set of sentences for which multiple outside annotators agree with the original NUCLE annotations. I begin with 341 randomly selected essays in the NUCLE corpus. From these, I looked for sentences that have been marked as having exactly one redundant region (it may contain one or more words). There are 863 such sentences. For each sentence, I ask five outside annotators to identify the most redundant phrase in the sentence independent of the NUCLE annotations. Specifically, I rely on the Amazon Mechanical Turk crowd-source service to provide multiple annotators for this task. Although they are not professional linguists or language tutors, the "Turkers" have shown to be helpful annotators for many NLP problems (Callison-Burch, 2009; Callison-Burch and Dredze, 2010; Tetreault et al., 2010b; Akkaya et al., 2010). Figure 9 shows a screen-shot of the annotation gathering interface used by the Turkers.

For a sentence to be included in my "clear-cut" corpus, a majority of the five Turkers must agree with the original NUCLE annotation (Figure 10). In other words, I retain the sentence if and only if at least three Turkers agree on the same span as the NUCLE annotator. This type of

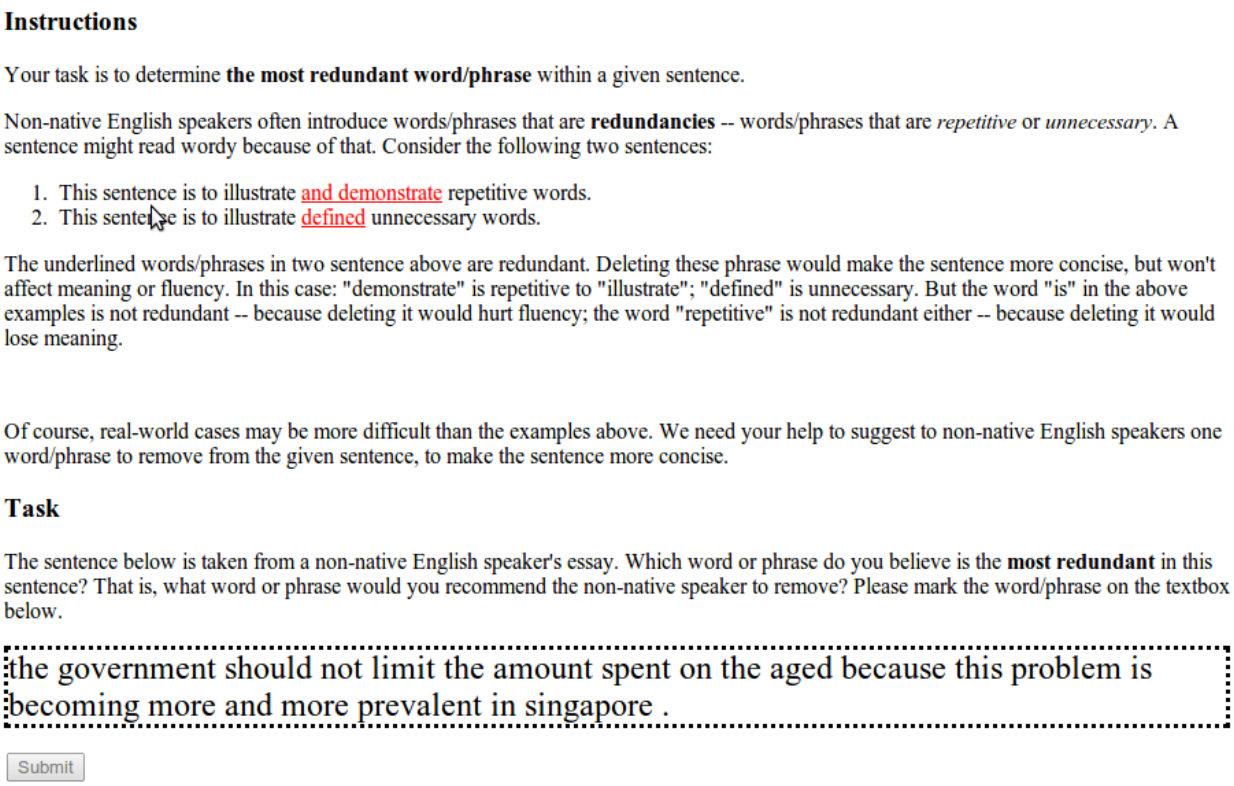


Figure 9: Word level redundancy annotation. Annotators are asked to pick the most redundant word/phrase within a given sentence.

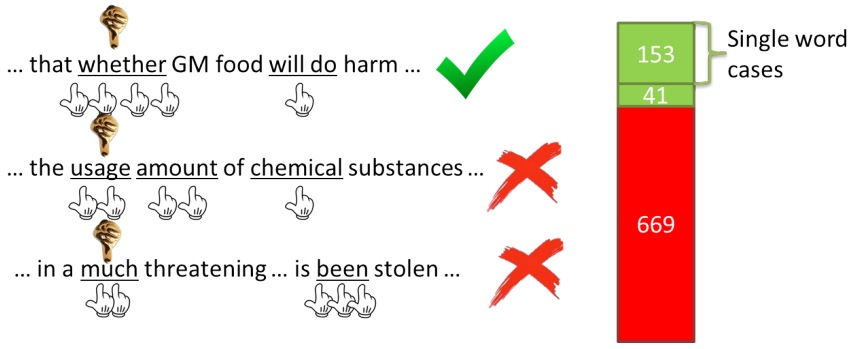


Figure 10: Using five Turkers' annotations to filter unclear cases in NUCLE annotations. For each sentence, the word that the golden finger points at is the most redundant word picked by the NUCLE annotator. The white fingers are the Turkers' annotations. A sentence is kept if and only if the majority vote of the Turkers matches the NUCLE annotation.

aggregation by majority voting has been used in previous work (Tetreault et al., 2010b) and has been shown to improve the annotation reliability (Snow et al., 2008; Sheng et al., 2008).

In the end, 669 sentences are filtered out, leaving 194 sentences in the clear-cut corpus. Table 4 shows a distribution based on the length of the redundant phrase. This distribution is consistent with the full corpus distribution shown in Figure 5. Out of the clear-cut corpus, 153 instances contain single word redundancy. These 153 sentences are used for evaluation and development in my experiments. Among the 669 filtered sentences, 449 (67.12%) are cases in which $\geq$2 Turkers agree on a different span to be most redundant. This supports my intuition that existing corpora do not mark redundant phrases consistently.

4.3.3 Features for Describing Word-Level Redundancy

In terms of feature development, I try to capture both shallow patterns and deeper reasons of redundancies. In addition to a wide range of features commonly used for many NLP classification problems, I also look for measures that quantify how much a word contributes to the meaning and fluency of the sentence.

Length	Percentage (%)
1	78.87% (153/194)
2	11.34% (22/194)
3	3.61% (7/194)
4	3.09% (6/194)
5	2.06% (4/194)
6	1.03% (2/194)

Table 4: Length distribution of redundant parts in my filtered corpus.

4.3.3.1 Features: Shallow Redundancy Patterns Although shallow redundancy patterns may not be a very strong indicator of redundancy in-and-of themselves, they work together with other features to improve the overall accuracy of the classifier. They fall into three main categories.

CERTAIN WORD TYPES ARE MORE LIKELY TO BE REDUNDANT. For example, function words such as "for" are more likely to be redundant than content words such as "citizen". Also, certain collocations, such as "request for" can suggest redundancies. I use each word's identity **unigram**, and its word type **POS** as features. I also introduce **special-case** features to indicate special word types including connectors (e.g. however), contrast words (e.g. yet), negation words (e.g. but), or if the word occurs at the beginning of a sentence. Beyond single word features, I include **POS-bigram** to capture collocation patterns.

REDUNDANT WORDS APPEAR AT NON-IMPORTANT LOCATIONS For example, modifiers (e.g. *usage amount*) are more likely to be redundant than central words (e.g. *chemical substances*). I introduce features based on the word's position in the sentence's fluency flow **LM** and syntax structure **sig-score**.

LM Deleting a redundant word does not hurt the sentence's fluency. I consider relying on large scale language models to decide redundancy, by calculating the log-likelihood of the whole sentence after discarding the given word/phrase. I in particular use a trigram language model. A higher probability indicates a higher fluency.

sig-score Redundant words may reside shallower than central words. I borrow **sig-score** (Clarke

and Lapata, 2007) to quantify the word's importance based on its location in sentence's syntactic structure. This measure was introduced in sentence compression to account for whether one word w_i is capturing the gist of a sentence. [1].

$$I(w_i) = -\frac{l}{N} \cdot f_i \log \frac{F_a}{F_i}$$

f_i and F_i are the frequencies of w_i in the current document and a large corpus respectively; F_a is the number of all word occurrences in the corpus; l is the number of clause constituents above w_i; N is the deepest level of clause embeddings. This measure assigns low scores to document specific words occurring at deep syntax levels.

REDUNDANT WORDS MAY BE DISCARDED AFTER WE AGGRESSIVELY POST-EDIT THE SENTENCE. Consider $\mathbf{Ex}_4$, "substances" is less likely to be modified than "usage" if we aggressively post-edit the sentence. Inspired by Madnani et al. (2012a), an MT system be used for aggressively post-editing sentences with the help of large scale language models. This method in particular first translates the sentence into a foreign language, and then back to English, using Google translate. I employ this method for measuring redundancies, by passing the input sentence through this **round-trip**, and measure whether each word has disappeared. I determine if one word disappeared in two ways

1. **extract word match**: one word is considered disappeared if the same word does not occur in the round-trip.
2. **aligned word**: I use the Berkeley aligner (DeNero and Klein, 2007) to align original sentences with their round-trip translations. Unaligned words are considered to have disappeared.

The above redundancy measures can be categorized into *binary feature sets* and *numerical feature sets*, which I summarize in Table 5. A binary feature set contains a set of features each indicating whether one condition holds in the certain context (e.g. **unigrams**, **POS**). Numerical features are single features that, in my case, measure a certain aspect of redundancy (e.g. **sig-score**, **LM**). Although both types of features can work in our machine learning systems, the numerical features can also be used in isolation. To gain more insight into these individual numerical features, I compare them in isolation during experiments (Section 4.3.4).

[1] I extend this measure, which was only defined for content words in Clarke and Lapata (2007), to include all English words.

Type	Redundancy measure
Binary measures	**unigram** **POS** **special-case** **POS-bigram**
Numerical measures	**LM** **round-trip** (both **extract word match** and **aligned word**) **sig-score**

Table 5: List of numerical and binary features proposed for word-level redundancy detection.

4.3.3.2 Features: Contributions to Meaning and Fluency In addition to the features introduced above, I also want to develop a quantitative measure that has a more theoretical underpinning in terms of describing word redundancy. In particular, I explore the use of machine translation to evaluate the semantic content of words. In the rest of this subsection, I present a probabilistic formulation of redundancy in terms of translations. I also make some approximations/deductions that lead to two efficient redundancy measures.

I consider a word or a phrase to be redundant if deleting it results in a fluent English sentence that conveys the same meaning as before. For example, in sentence shown in Figure 11, among the three circled words, "just" is more redundant because deleting it hurts nether fluency nor meaning. This is because discarding "not" would flip the sentence's meaning; discarding "the" would lose a necessary determiner before a noun. In contrast, discarding "just" would hurt neither fluency nor meaning. It is thus considered to be more redundant.

Therefore, my computational model needs to consider each word's contribution to both fluency and meaning. A relatively straightforward measure of fluency is to use a language model in some way; but choosing a measure for meaning is less straightforward.

MEASURING CONTRIBUTION TO MEANING WITH TRANSLATION AND ALIGNMENTS

The key insight in my formulation is that: a sentence's translation is a good approximation of its meaning (Hermet and Désilets, 2009; Madnani et al., 2012a). The alignment between two

Figure 11: Roles of each word's contribution to fluency and meaning in determining whether it is redundant. Among the three circled words, "just" is more redundant because deleting it hurts nether fluency nor meaning. Deleting "not" or "the" would hurt either the sentence's meaning or fluency.

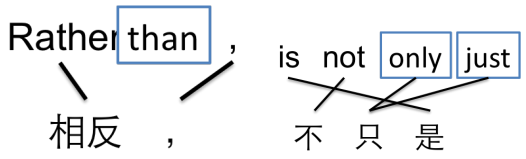
sentences can help reflect each word's contribution to meaning. Figure 12 illustrates words' contribution to meaning. In those two examples, each sub-graph visualizes a sentence: English words in the top row are aligned to their translations in the bottom row. Each translated word roughly represents a meaning component. The knowledge of which word corresponds to which meaning component helps in evaluating an English word's contribution. In particular, if a word does not align with any translated word, deleting it would not affect the overall sentence; if several words align with the same translated word, then deleting some of them might not affect the overall sentence either. Also, deleting a more semantically meaningful word (or phrase) is more likely to cause a loss of meaning of the overall sentence (e.g. *uncertain* v.s. *the*).

PROBABILISTIC FORMULATION

I propose a probabilistic model that computes a single value for both fluency judgment and meaning preservation – the log-likelihood that after deleting a certain word or phrase of a sentence, the new sentence is still fluent and conveys the same meaning as before. This value reflects my definition of redundancy – the higher this probability, the more redundant the given word/phrase is.

More formally, suppose an English sentence e contains l_e words: $e = e_1 e_2 \dots e_{l_e}$; after some word $e^k (1 \leq k \leq l_e)$ is deleted from e, we obtain a shorter sentence, denoted as e^k_-. I wish to compute the quantity $R(k; e)$, the chance that the word e_k is redundant in sentence e.

I propose a probabilistic model to formalize this notion. Let M be a random variable over some meaning representation; $\Pr(M|e)$ is the likelihood that M carries the meaning of e. If the word e_k is redundant, then the new sentence e^k_- should still express the same meaning; $\Pr(e^k_-|M)$



(a) Unaligned English words are considered redundant.

(b) Multiple English words aligned to the same meaning unit. These words are considered redundant.

Figure 12: Configurations my system consider as redundant. In each figure, the shaded squares are the words considered to be more redundant than other words in the same figure.

computes the likelihood that the after-deletion sentence can be generated from meaning M.

$$\begin{aligned} & R(k;e) \\ = \; & \log \sum_{M=m} \Pr(m|e) \Pr(e^k_-|m) \\ = \; & \log \sum_{M=m} \frac{\Pr(m|e) \Pr(e^k_-) \Pr(m|e^k_-)}{\Pr(m)} \\ = \; & \log \Pr(e^k_-) + \log \sum_{M=m} \frac{\Pr(m|e^k_-) \Pr(m|e)}{\Pr(m)} \\ = \; & \mathrm{LM}(e^k_-) + \mathrm{AGR}(M|e^k_-, e) \end{aligned} \tag{4.1}$$

The first term $\mathrm{LM}(e^k_-)$ is the after-deletion sentence's log-likelihood, which reflects its fluency. I calculate the first term with a trigram language model (LM).

The second term $\mathrm{AGR}(M|e^k_-, e)$ can be interpreted as the chance that e and e^k_- carry the same meaning, discounted by "chance agreement". This term captures meaning preservation.

The two terms above are complementary to each other. Intuitively, LM prefers keeping common words in e^k_- (e.g. *the, to*) while AGR prefers keeping words specific to e (e.g. *disease, hypertension*).

To make the calculation of the second term practical, I make two simplifying assumptions.

Assumption 1 A sentence's meaning can be represented by its translations in another language; its words' contributions to the meaning of the sentence can be represented by the mapping between the words in the original sentence and its translations .

Note that the choice of translation language may impact the interpretation of words' contributions. I will discuss about this issue in my experiments (Section 4.3.4).

Assumption 2 Instead of considering all possible translations f for e, my computation will make use of the most likely translation, f_*.

With the two approximations:

$$\begin{aligned} \mathrm{AGR}(M|e_{-}^{k}, e) &\approx \log \frac{\Pr(f_*|e_{-}^{k}) \Pr(f_*|e)}{\Pr(f_*)} \\ &= \log \Pr(f_*|e_{-}^{k}) + C_1(e) \end{aligned}$$

(I use $C_i(e)$ to denote constant numbers within sentence e throughout the chapter.)

Therefore, my redundancy measure boils down to calculating the sum of a language model score $\mathrm{LM}(e_{-}^{k})$ and a translation probability $\log \Pr(f_*|e_{-}^{k})$. Intuitively, $\log \Pr(f_*|e_{-}^{k})$ measures how likely deleting a word e_k does not change the meaning. I use an example to illustrate this idea. Consider e being "It is not only just good.", its translation f^* being "它不只是好". In this case, after deleting $e_k =$ "just", the following translation probability would remain high.

$$\log \Pr(f_*|e_{-}^{k}) = \log \Pr(\text{它不只是好}|\text{"It is not only good."})$$

This suggests that "just" is more redundant.

I now rely on a statistical machine translation model to approximate the translation probability $\log \Pr(f_*|e_{-}^{k})$.

APPROXIMATIONS

One naive way of calculating this probability measure is to consult the MT system. This method, however, is too computationally expensive for one single input sentence. For a sentence of length n, calculating the redundancy measure for all words in it would require issuing $O(n)$ translation queries. I propose an approximation that instead calculates the difference of translation probability caused by discarding e_k, based on an analysis on the alignment structure between e and f_*. I show the measure boils down to counting the expected number of aligned words for e_k, and weighting it by e_k's unigram probability. This method requires one translation query, and $O(n)$ queries into a language model, which is much more suitable for practical applications. My method also sheds light on the role of alignment structures in the redundancy detection context.

Note that Statistical Machine Translation systems (Brown et al., 1993; Och, 2003) often compute the translation probability $\Pr(f^*|e_{-}^{k})$ in roughly two steps:

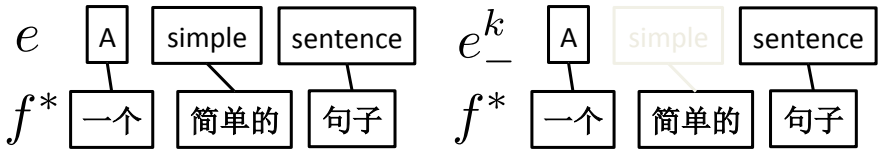


Figure 13: Illustration of Approximation 1. Sentence alignments normally won't be affected before/after deleting words (e.g. "simple") from the source sentence.

1. Compute the alignment between the two sentences.
2. Calculate the translation probability given the alignment.

I approximate the two steps separately

1. I use the alignment between e and f^* to approximate the alignment between e^k_-. The key insight is that the alignment structure a between e^k_- and f_* would be largely similar with the alignment structure between e and f_*. I illustrate this notion in Figure 13. Note that after deleting "simple" from the source sentence, the alignment structure remains unchanged elsewhere. Also, "简 单 的", the word once connected with "simple", can now be seen as connected to a blank.
2. I use IBM Model 1 (Brown et al., 1993) to calculate the translation probability given the alignment. The model formalizes that each word contributes to its alignment. As illustrated in Figure 14, this helps us to capture that deleting unaligned words (e.g. "than") or repetitively aligned words (e.g. "only") does not hurt the probability.

I formalize the two approximations below.

Approximation 1 Let $\Pr(a|f,e)$ be the posterior distribution of alignment structure between sentence pair (f,e). I formalize the similarity between the alignment structures by assuming the KL-divergence between their alignment distributions to be small.

$$D_{\mathrm{KL}}(a|f_*,e;a|f_*,e^k_-) \approx 0$$

Approximation 2 I will use IBM Model 1 (Brown et al., 1993) to calculate $\log \Pr(f_*|e^k_-,a)$. IBM Model 1 is one of the earliest statistical translation models. It helps us to compute $\log \Pr(f_*|e^k_-,a)$

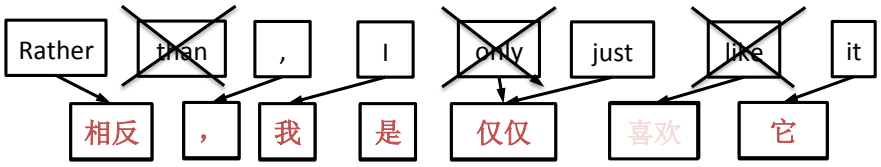


Figure 14: Using IBM model to capture each word's contribution to its alignment. Each English word contributes to the Chinese word it aligns with. Deleting an unaligned English word, e.g. "than", would not change the translation; deleting a repetitively aligned word, e.g. "only", would not change the translation; deleting a singly aligned word, e.g. "like", would risk losing its aligned Chinese word.

by making explicit how each word contributes to words it aligns with. In particular, to compute the probability that f is a translation of e, $\Pr(f|e)$, IBM Model 1 defined a generative alignment model where every word f_i in f is aligned with exactly one word e_{a_i} in e, so that f_i and e_{a_i} are word level translations of each other.

A New Redundancy Measure After mathematical deductions in Appendix A.2, I get the proposed redundancy measure shown in Equation 4.2.

$$R(k;e) \approx \underbrace{\text{LM}(e^k_-)}_{\text{Fluency without } e_k} + \underbrace{A(k)\log\Pr(e_k)}_{\text{meaning redundancy}} + \underbrace{C(e)}_{\text{Constant number}} \tag{4.2}$$

The proposed measure contains three components.

1. $\text{LM}(e^k_-)$ captures e_k's redundancy in terms of its contribution to the sentence's fluency. It is the log-likelihood of the input sentence after discarding e_k. When deleting e_k does not hurt fluency, this component will assign a high score.
2. $A(k)\log\Pr(e_k)$ captures e_k's redundancy in terms of its contribution to sentence's meaning. It is the multiplication of two parts. $A(k)$ is the number of words aligned with e_k in f^*. Figure 15 illustrates how A is calculated. A helps us to assign high scores to words that receive less alignments. $\log\Pr(e_k)$ is the unigram probability of e_k. It helps to formalize that rare words are often less redundant.

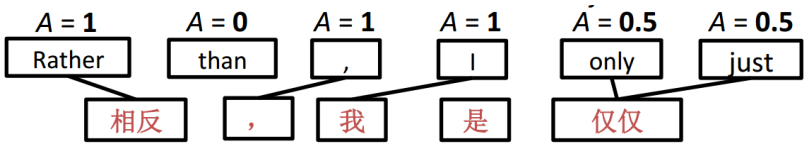


Figure 15: Illustrating how $A(k)$ is calculated for input sentence e =“Rather than , I only just . . . ”, and its translation f^* =“相反，我是仅仅……”. When a word aligns with one single word in f^* (e.g. “rather”), $A = 1$; when it aligns with nothing in f^* (e.g. “than”), $A = 0$; when two words aligns with the same word in f^* (e.g. “only” and “just” both align with “仅仅”), $A = \frac{1}{2}$.

3. $C(e)$ is a constant number for the input sentence. For redundant word detection, it can be disregarded.

The **LM** component is already included as one of the many shallow features. The derivations of this section provide some theoretical justification for its inclusion. Moreover, the equation suggests that our feature set should also include the second component, denoted as **MeanR**. When estimating the alignment probabilities $\hat{A}(k)$ for **MeanR**, I smooth the alignment result $A(k)$ from Google translation using Dirichlet-smoothing, where I set $\alpha = 0.1$ empirically based on experiments in the development dataset.

$$\hat{A}(k) = \frac{A(k) + \alpha}{\sum_{1 \le i \le l_e} A(i) + l_e \alpha}$$

4.3.4 Experiments

I conduct experiments to investigate these questions:

1. How accurately can the most redundant word of a sentence be detected through automatic means?
2. Does a sentence’s translation serve as a reasonable approximation for its meaning?
3. If so, does the choice of the pivot language matter?
4. How do the potentially conflicting goals of preserving fluency versus preserving meaning impact the definition of a redundancy measure?

To answer the first question, I experiment with different feature sets to evaluate their abilities to capture redundancy (each feature set constitutes a different "system"). I calculate a system's accuracy as the percentage of times that its choice of the most redundant word agrees with that of the human annotators. The systems are compared against a baseline that picks a random word as its answer.

To answer the second and fourth question, I use the translation-based measures directly (either in isolation or as linearly interpolated combinations) instead of as a part of a larger feature set for the classifier.

To answer the third question, I compare system performance while varying among 52 *pivot languages*[2].

4.3.4.1 Data and Tools The experiment uses the set of 153 sentences I collected (see Section 4.3.2) for evaluation and development. Among the 153 sentences, I randomly pick 91 as test cases; I use the rest 62 sentences for development and training purposes. However, since 62 sentences are not enough to generate a sizable training corpus, I also included some not as clear-cut sentences for training. In particular, I took the 412 single word cases extracted from the 669 sentences that we had filtered out. In these sentences, although annotators do not all agree on the same word as being the most redundant, the original NUCLE annotation is still valuable: the marked word is still more redundant than most other words in the sentence. Since each sentence contributes as many training examples as there are words, most of the instances generated from the not as clear-cut sentences are still valuable training examples.

The language model used for this experiment is a trigram model trained using the SRILM toolkit (Stolcke, 2002) on the Agence France-Presse (afp) portion of the English Gigawords corpus (Consortium et al., 2003). I use Google translate's research API [3] to gather translations.

[2]These include: Albanian (sq), Arabic (ar), Azerbaijani (az), Irish (ga), Estonian (et), Basque (eu), Byelorussian (be), Bulgarian (bg), Icelandic (is), Polish (pl), Persian (fa), Boolean (language ((Afrikaans) (af), Danish (da), German (de), Russian (ru), French (fr), Tagalog (tl), Finnish (fi), Khmer (km), Georgian (ka), Gujarati (gu), Haitian (Creole (ht), Korean (ko), Dutch (nl), Galician (gl), Catalan (ca), Czech (cs), Kannada (kn), Croatian (hr), Latin (la), Latvian (lv), Lao (lo), Lithuanian (lt), Romanian (ro), Maltese (mt), Malay (ms), Macedonian (mk), Bengali (bn), Norwegian (no), Portuguese (pt), Japanese (ja), Swedish (sv), Serbian (sr), Esperanto (eo), Slovak (sk), Slovenian (sl), Swahili (sw), Telugu (te), Tamil (ta), Thai (th), Turkish (tr), Welsh (cy), Urdu (ur), Ukrainian (uk), Hebrew (iw), Greek (el), Spanish (es), Hungarian (hu), Armenian (hy), Italian (it), Yiddish (yi), Hindi (hi), Indonesian (id), English (en), Vietnamese (vi), Simplified Chinese (zh-CN), Traditional Chinese (zh-TW).

[3]http://research.google.com/university/translate/

4.3.4.2 Experiment Results

My experimental results are presented in Table 6 and Figure 16. In Table 6, I compare building redundancy detectors using different measures for the same pivot language – French. In Figure 16, I compare using different pivot languages in the system using the linear combination of two of the measures **LM+MeanR**.

These results provide some quantified answers our experimental questions. First, the best feature combination results in a detector that identifies the most redundant words in the test sentences with a 37% accuracy. This is seven times more accurate than the random baseline. While the raw accuracy percentage is not yet high enough for a downstream application, it suggests that the notion of redundant word is not random. This validates my hypothesis that there are regular patterns that describe redundant words.

Second, the redundancy measure that I proposed (**LM+MeanR**) does get at the deeper reasons for word-level redundancy. Using it alone without learning (i.e., **LM** and **MeanR** are added linearly) we are able to pick the most redundant word with an accuracy of 26.37%. Moreover, by excluding both **MeanR** and **LM** as features, the classifier accuracy drops to 18.68%.

Third, I find that the choice of pivot language does make a difference. Initial experimental result suggests that the system tends to achieve higher redundancy detection accuracy when using translations of a language more similar to English. In particular, when using European languages (e.g. German (de), French (fr), Hungarian (hu) etc.) as pivot, the system performs much better than using Asian languages (e.g. Chinese (zh-CN), Japanese (ja), Thai (th) etc.). One reason for this phenomenon is that the default Google translation output in Asian languages (as well as the alignment between English and these languages) are organized into characters, while characters are not the minimum meaning component. For example, in Chinese, "解释" is the translation of "explanation", but the two characters "解" and "释" mean "to solve" and "to release" respectively. In the alignment output, this will cause certain words being associated with more or less alignments than others. In this case, the number of alignments no longer directly reflect how many meaning units a certain word helps to express. To confirm this phenomenon, I tried improving the system using Simplified Chinese as the pivot language by merging characters together. In particular, I applied Chinese tokenization (Chang et al., 2008), and then merged alignments accordingly. This raised the system's accuracy from 18.68% to 29.67%.

Fourth, a closer look at the system's output reveals **LM** and **MeanR**'s different roles in iden-

	model	accuracy
Single measures	Random	5.49%
	LM	15.38%
	round-trip (aligned word)	6.59%
	round-trip (exact word match)	6.59%
	sig-score	5.49%
	MeanR	5.49%
	LM+MeanR	26.37%
Incorporating multiple measures in MaxEnt	Binary	18.68%
	Binary $\cup$ LM	25.27%
	Binary $\cup$ LM $\cup$ round-trip (aligned word)	25.27%
	Binary $\cup$ LM $\cup$ round-trip (exact word match)	23.08%
	Binary $\cup$ LM $\cup$ sig-score	17.58%
	Binary $\cup$ LM+MeanR	37.36%

Table 6: Comparison of different redundancy measure combinations for our word-level task.

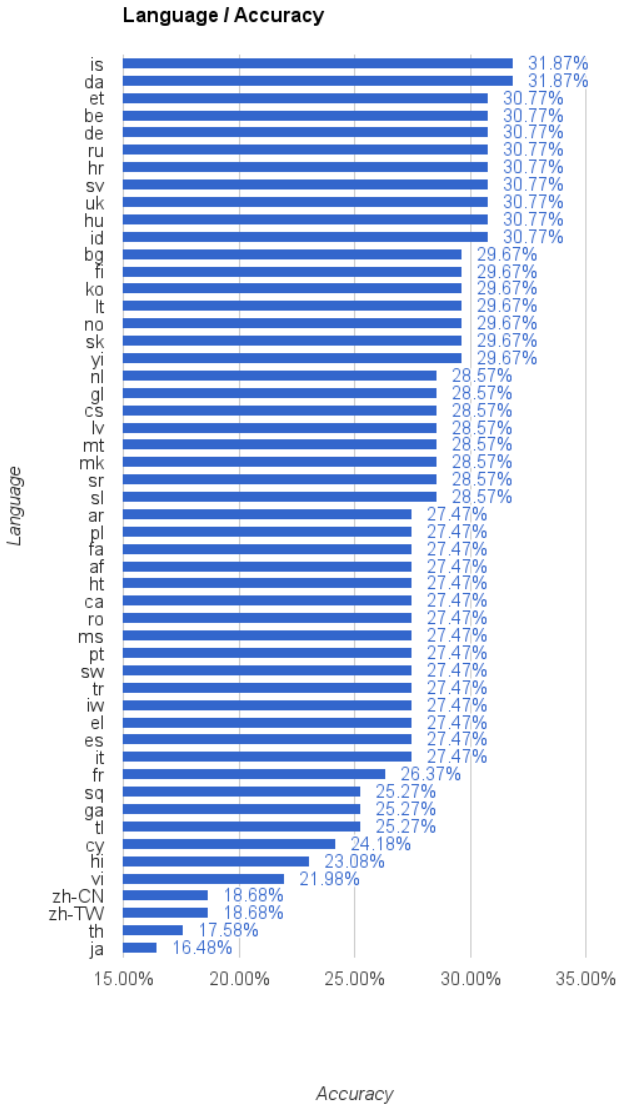


Figure 16: List of **LM+MeanR** prediction accuracies using different pivot languages. I only consider non-English languages that are supported by Google translator. In general, European languages helps the system to achieve better accuracies, compared with Asian languages.

tifying redundancies. Below are examples where the linear add-up correctly identifies the most redundant word.

1. In "illiteracy often limits the economical growth of a nation where knowledge intensive industries are much highly valued . ":
   a. The **LM** measure assigns the highest score to "economical".
   b. The **MeanR** measure assigns the highest score to "the".
   c. By adding them up, **LM+MeanR** assigns the highest score to "much".
2. In "to conclude , not only are the renewable fuels a lot more eco-friendly , but they also have are widely and cheaply available . "
   a. The **LM** measure assigns the highest score to "the".
   b. The **MeanR** measure assigns the highest score to "eco-friendly".
   c. By adding them up, **LM+MeanR** assigns the highest score to "have".

In general, I find that the language model component prefers to detect rare content words, while the alignment analysis component prefers to detect function words. However, the English language model and the alignment analysis result can build on top of each other when we analyze the redundancies.

4.4 TASK 2: SENTENCE-LEVEL REDUNDANCIES

Perhaps more so than deciding whether one word is more redundant than another within a sentence, ESL students are likely to ask: Does my sentence sound concise enough? For the language tutors as well, a system that identifies wordy sentences is arguably more time-saving than one that identifies which word to remove within a sentence (because it helps the tutor to zero in on a few problematic sentence out of a much longer essay).

From a research perspective, this task seems like a natural extension of the previous task. If I have a reliable word-level redundancy measures, it seems like I ought to be able to compute some sort of global redundancy measure for the sentence based on the value of the most redundant word. However, there are some significant challenges. One is the lack of a reliable dataset. Another is in

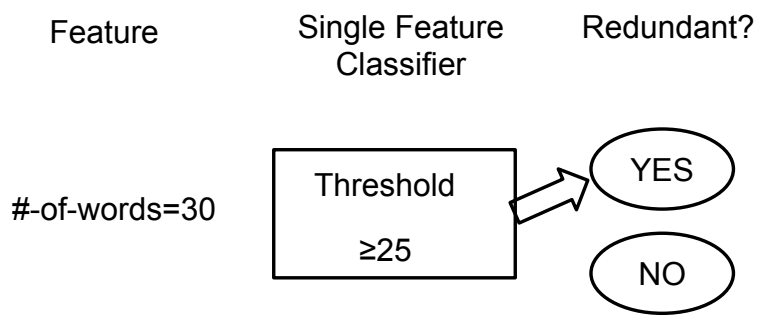


Figure 17: Single feature classifier for sentence level redundancy detection. The classifier compares the feature value in the input sentence with a threshold, to decide whether to output "yes" or "no". The decision threshold is learned to maximize prediction accuracy on the training set.

managing the interaction between local word-level features (such as the redundancy scores of each word) and global sentence-level features (such as the length of the sentence).

I address these challenges in my investigation of sentence-level redundancies. First, I develop a corpus based on aggregate annotators. Second, I develop a binary classifier making use of measures including both global and local features. From the experimental results, I find that:

1. among current individual measures, sentence length and unigrams best correlate with sentence-level redundancy judgments.
2. our combined measure achieves 70% accuracy on the balanced dataset I developed.

4.4.1 System Framework

Sentence-level redundancy detection is a natural fit with binary classification. Given a sentence, the classifier extract features from it and predict whether it is redundant or not. Once again, I use a MaxEnt classifier. As a point of comparison, I also consider a threshold-based single feature classifier (Figure 17).

4.4.2 Corpus Collection

Corpus Development for sentence-level redundancy detection is more challenging than the word-level case. Human judgment in this case is arguably more subjective than the word-level case. People have different degree of tolerance for redundancy; minimalists want bare-bone sentences, others might withstand elaborate phrases with a stream of semantically similar adjectives as long as the sentence is grammatical. Moreover, even though redundancy is a common writing problem, it does not occur (or, at least, is not marked) in most student sentences. Therefore, there may be a strong imbalanced label problem, where the "not redundant" cases overwhelm the "redundant" ones. Another related issue is that while there are some regular patterns to redundant phrases, it is much harder to characterize all the ways in which a sentence does not have redundant phrases.

With these issues in mind, I set out to construct a balanced corpus in which half the sentences are clearly not redundant and half of the sentence clearly contains a redundant phrase. Since we already have 153 instances of sentences with one redundant phrases (Section 4.3.2), I need to identify 153 sentences that most people would agree to not contain any redundant phrases.

My collection procedure is similar to before. I begin with a set of 1000 sentences from NUCLE that do not have any error annotations. Then, each sentence is read by five Turkers. I use a slightly modified annotation interface from the earlier word-level version (Figure 9); the only difference is that it now allows Turkers to specify that no word is redundant (Figure 18). To minimize the impact of the greater variance in human judgment, I keep a sentence if and only if all five annotators agree it is not redundant. In the end, 215 sentences have been kept. To match the size of the redundant sentence set, I randomly pick a 153 sentence subset.

4.4.3 Features: Sentence-Level Redundancy Measures

I compare the following sentence-level measures:

- Redundant sentences tend to be longer. So I use the number of words **#-words**, and the number of characters **#-char** measure the sentence length.
- Certain words may correlate with redundancy. For example, when a sentence has *because*, we may need to remove words to reduce its complexity. I use **unigrams** as binary features to capture this notion. More specifically, I create one feature for every vocabulary word, indicating

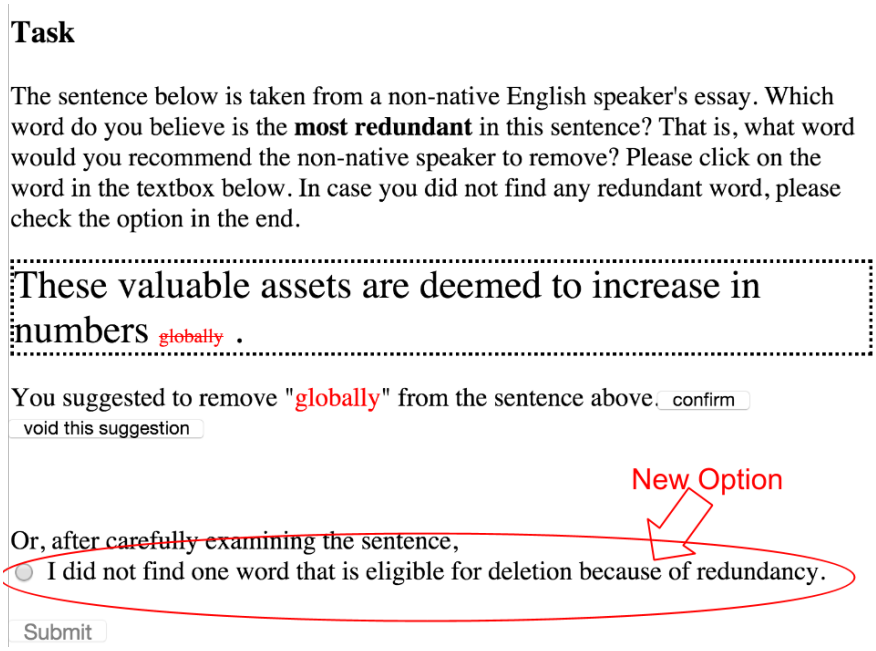


Figure 18: Amazon Mechanical Turk annotation interface for the sentence-level redundancy detection. Compared with word-level redundancy detection, I added an additional option to say that the sentence does not contain redundancies at all.

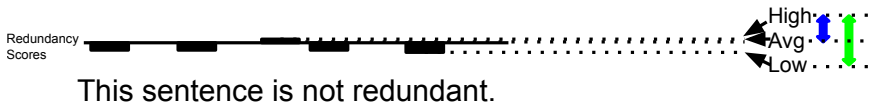


Figure 19: We expect redundancy scores to have lower variations in non-redundant sentences.

whether the sentence contains that word.

- Redundant sentences may have a smaller meaning/length ratio. I develop two measures based on different ways to measure "meaning"

  avgalign The average number of aligned words to the sentence's translation. As discussed in Section 4.3.3.2, a sentence's translation can be used as a representation of its meaning. Then, the number of aligned words roughly measures one word's contribution to meaning.

  perplexity The per-word perplexity according to a trigram language model. Meaning is information. In language modeling, perplexity is a measure of the *information* one word brings (Brown et al., 1992).

- Some part is significantly more redundant than other parts. As illustrated in Figure 19 and Figure 20, normally we expect some words to "stand out" in a redundant sentence. In terms of word level redundancy scores, we would then expect a bigger gap between some words' scores and others'. To formalize it, given an English sentence, containing n words: $e = e_1 \dots e_n$. Suppose the word level redundancy score for e_i is $R(e_i)$. I consider two measures for the gaps:

$$\begin{aligned} \textbf{High} - \textbf{Avg}(e) &= \max_i(R(e_i)) - \frac{\sum_i(R(e_i))}{n} \\ \textbf{High} - \textbf{Low}(e) &= \max_i(R(e_i)) - \min_i(R(e_i)) \end{aligned}$$

Here, the word-level redundancy score $R()$ is computed using Fluency+Meaning (Section 2).

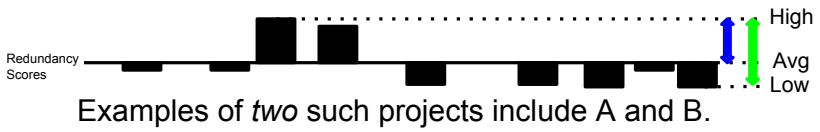


Figure 20: We expect redundancy scores to have higher variations in redundant sentences.

4.4.4 Experiment

I conducted experiments to answer the following questions:

1. How well can computers predict if a sentence contains redundancies?
2. Which of our measures best predicts sentence-level redundancies?
3. Does our word-level redundancy measures generalize to sentence-level?

To determine how well the various proposed measures can detect sentence-level redundancies, I compare classifiers trained with different subsets of features. Each trained classifier is evaluated on its prediction accuracy on the test set. The experiment is carried out with a ten-fold cross validation scheme over 91% of the balanced corpus I collected (I retained 9% of the corpus for development purposes). As a future work, we may consider incorporating noisy examples into training data (Section 4.3.4.1).

As before, the trigram language model is trained on English Gigawords. I use Google translate into Simplified Chinese to gather translation related features. Similar to Section 4.3.4.2, I tokenize the Chinese translation, and merge alignments accordingly. The reasons for picking Simplified Chinese are two-fold: (1) it is my native language, using which eases system development (2) as shown previously, this configuration yields among-the-top word-level redundancy detection accuracy.

Table 7 summarizes the current top-performing models.

Overall, a trained classifier does beat random baseline. The best feature combination achieves an average of 70% accuracy on the balanced corpus. However, of all the individual redundancy measures we have considered, **unigram** and the length related ones are better predictors than those

Model	Accuracy
Random	50%
MaxEnt(unigram+#-char)	70%
MaxEnt(all features)	**69%**
MaxEnt(unigram+#-words)	69%

Table 7: Model performance for sentence level redundancy detection. The top performing model predicts with a 70% accuracy on our balanced dataset.

based on our word-level redundancy scorer. Table 8 compares the performance of each feature individually.

While the direct use of word-level redundancy features(**High-Avg**) helps, our measures based on them do not directly add to sentence-level redundancy prediction accuracy, as shown in Table 9. There are many cases where the variance in word-level redundancy do not correlate with sentence-level redundancies. Take $\mathbf{High - Avg}$ as an example:

- A high score suggests it is more acceptable to discard one word than the others. But in Figure 21, all the words in the sentence are non-redundant. Even deleting "in", which computes the highest score, would heavily affect the sentence's grammatical structure.
- A low score suggests discarding every word is equally acceptable. But in Figure 22, many words can be considered redundant, such as "often", or the entire phrase "most of the times".

4.5 RELATED WORK

Compared to other ESL errors, detecting redundancies has not been as thoroughly studied. Two major challenges include the subjectivities in the judgment of redundancies, and the wide variety of redundant words/phrases. Unlike mistakes that violate the grammaticality of a sentence, some redundancies do not "break" the sentence. Determining which word or phrase is redundant may be a stylistic question; it is more subjective, and sometimes difficult even for a native speaker.

Model	Accuracy
MaxEnt(unigram)	**67%**
Single(#-char)	**63%**
Single(High-Avg)	61%
Single(#-words)	59%
Single(High-Low)	56%
Single(avgalign)	56%
Single(perplexity)	55%

Table 8: Top performing sentence level redundancy measures.

Model	Accuracy
MaxEnt(unigram)	67%
MaxEnt(unigram+High-Avg)	68%
MaxEnt(unigram+High-Low)	68%

Table 9: Fluency/meaning features do not add to prediction accuracy during the sentence-level task.

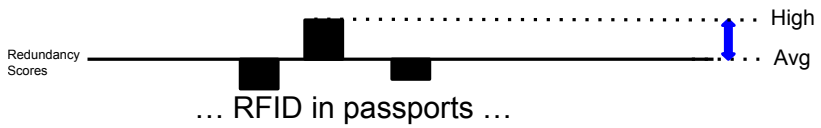


Figure 21: False positive example. In this non-redundant sentence, there is actually a big variance in word-level redundancy scores. But every of these words are non-redundant.

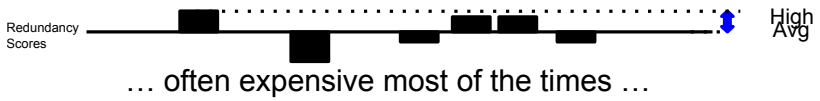


Figure 22: False negative example. This sentence contains many redundant words/phrases, so the variance in word-level redundancy scores is low.

Moreover, redundancies cannot be confined to some small fixed set of phrases (although some phrases are well known to be redundant: e.g., "ask a question"[4]).

Outside of GEC research, the most related research areas are sentence compression and sentence simplification, which also consider removing words from input sentences. However, their goals are somewhat different. Automated sentence simplification (Coster and Kauchak, 2011) systems aim at reducing the grammatical complexity of an input sentence. To illustrate the difference, let's consider the phrase "critical reception." A sentence simplification system might rewrite it into "reviews"; but a system that removes redundancy should leave it unchanged, because neither "critical" nor "reception" is extraneous. Moreover, consider the redundant phrase "are the things that" in $\mathbf{Ex_1}$. A simplification system would not need to change it because these words do not add complexity to the sentence. Sentence compression systems (Jing, 2000; Knight and Marcu, 2000; McDonald, 2006; Clarke and Lapata, 2007) aim to shorten a sentence while retaining the most important information and keeping it grammatically correct. This goal distinguishes these systems from ours, in two major aspects. First, sentence compression systems assume that the original sentence is well written; therefore retaining words specific to the sentence can be a good strategy (Clarke and Lapata, 2007). In the ESL context, however, even specific words can still be redundant (cf.$\mathbf{Ex_3}$). Although "usage" is specific to this sentence, it is redundant, because its meaning is already implied by "amount". Second, sentence compression systems try to shorten a sentence as much as possible, but an ESL redundancy detector should leave much of the input sentences unchanged.

[4]There are a few web lists of common redundant phrases: e.g., http://www.dailywritingtips.com/50-redundant-phrases-to-avoid/

4.6 CHAPTER SUMMARY

I have investigated the problem of automatically detecting redundant phrases in ESL writings. As the first study to take a closer look at this problem, I have defined two sub-tasks: (1) determining whether a word is more likely to be redundant than another; and (2) determining whether a sentence contains a redundant phrase or not. I have also constructed two corpora consisting of "clear-cut" examples for these tasks. For the first task, I have developed key features for measuring a word's potential of being redundant; they are based on translation models and language models. The best performing classifier achieves an accuracy of 37%. For the second task, I have investigated the roles of global features (based on sentence lengths) and local features (based on a word's potential for redundancy). The best performing classifier has a 70% accuracy, but it does not rely on those redundancy measures we have developed for the word-level detection task.

5.0 CORRECTION DETECTION IN ESL REVISIONS

5.1 CHAPTER OVERVIEW

When providing feedback to learners, a good practice is to suggest a fix, and also indicate the reason. Making sure that the reasons and fixes are provided in a consistent way can help reduce the learners' confusions (Fregeau, 1999). For example, in the sentence "He pick on a book", instead of suggesting that "pick on" needs to be replaced into "picks up", it would be better to indicate that two mistakes occurred in this sentence: one is a verb form error, the other is a preposition mis-usage:

1. To fix the verb form error, we should replace "pick" into "picks".
2. To fix the preposition mis-usage, we should replace "on" into "up".

Based on such feedback, we can also provide learners their error frequency statistics, which can help learners identify their deficiencies.

However, annotating in such a consistent way requires a great deal of effort from the language tutors. Language tutors have to isolate their corrections and label the reasons according to a pre-defined standard. This also requires the language tutors' continuous attention. When learners obtain feedback from online language exchange forums (e.g. lang-8.com), language partners often skip these steps – they edit inline directly. Recently, researchers have developed computer programs that are able to auto-complete some of these missing information. Swanson and Yamangil (2012) have proposed a system that compares the revised sentence with the original sentence to infer the locations of the errors being fixed, how they were fixed, and the rationales for the fixes. Figure 23 is an illustration of their system.

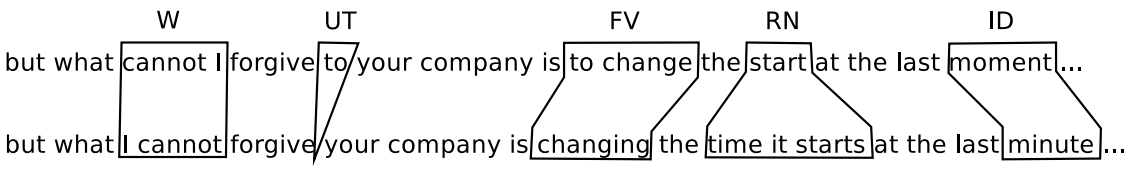


Figure 23: Detecting corrections from revisions. Swanson and Yamangil (2012)'s system detects individual corrections by comparing the original sentence with its revision, so that each correction addresses one error. Each polygon corresponds to one correction; the labels are codes of the error types. The codes follow the annotation standard in FCE corpus (Nicholls, 2003). In this example, *W* is incorrect Word order; *UT* is Unnecessary preposiTion; *FV* is wrong Verb Form; *RN* is Nnoun needs to be Replaced; *ID* is IDiom error.

S&Y's approach has two components: one to detect individual corrections within a revision, which they termed *correction detection*; another to determine what the correction fixes, which they termed *error type selection*. Although they reported a high accuracy for the error type selection classifier alone, the bottleneck of their system is the other component – correction detection. An analysis of their system shows that approximately 70% of the system's mistakes are caused by mis-detections in the first place. The major difficulty is ambiguities – there are often multiple ways to interpret how many corrections are performed. S&Y's approach relies on manually crafted heuristics developed on one corpus, which often fails to infer the correct interpretation. Also, since the heuristics are developed on one corpus, it is not clear whether it can generalize over different annotation code standards.

In this chapter, I answer the following questions

1. Can we build computer programs that more accurately interpret revisions?
2. How can computers handle ambiguities during correction detection? What is the major challenge in handling these ambiguities?
3. What computational models can help us handle the ambiguities better?
4. Can my computational models generalize over different coding standards?

I show empirically that a major challenge in correction detection is to determine the number of edits that address the same error. I propose to train a classifier to help determine which edits in a revised sentence address the same error in the original sentence. My trained classifier reduces mis-detection by 1/3, leading to significant improvement in the accuracies of combined *correction detection* and *error type selection*. I have conducted experiments across multiple corpora, indicating that the proposed merging model is generalizable.

5.2 CORRECTION DETECTION

Comparing a student-written sentence with its revision, we observe that each correction can be decomposed into a set of more basic edits such as word insertions, word deletions and word substitutions. In the example shown in Figure 23, the correction "*to change* ⇒ *changing*" is composed of a deletion of *to* and a substitution from *change* to *changing*; the correction "*moment* ⇒ *minute*" is itself a single word substitution. Thus, we can build systems to detect corrections which operates in two steps: (1) detecting the basic edits that took place during the revision, and (2) merging those basic edits that address the same error. Figure 24 illustrates the process for a fragment of the example sentence from above.

In practice, however, this two-step approach may result in mis-detections because there may be ambiguities. For example, the correction from *"a few other previous work"* to *"a few other studies"* can be interpreted in multiple ways. In Figure 25, I list three of them. Among the three interpretations, although it is easy for humans to agree the first two interpretations are more reasonable than the third one, it is not obvious to computers.

Mis-detections may be introduced from either steps. For example, to detect basic edits, Swanson and Yamangil applied the string edit distance algorithm (Levenshtein, 1966). Figures 26 gives an example of problems that might arise when applying edit distance. Basically, the edit distance algorithm only tries to minimize the number of edit operations. It does not care whether the edits make any linguistic sense. To detect the scope of the correction, Swanson and Yamangil applied a distance heuristic – basic-edits that are close to each other (e.g. basic edits with at most one word lying in between) are merged. Figure 27 shows cases for which the heuristic comes up with the

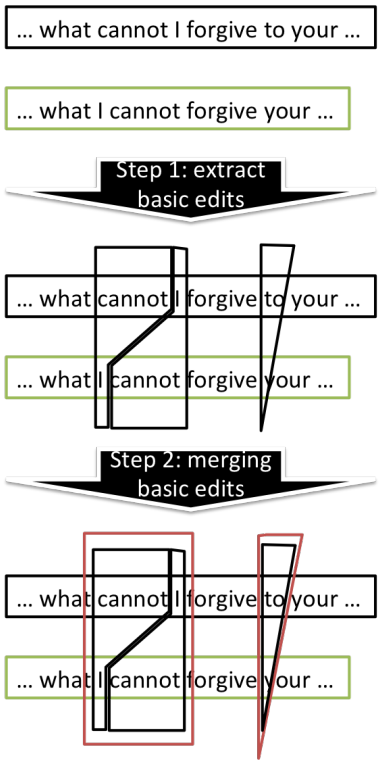


Figure 24: A portion of the example from Figure 23 undergoing the two-step correction detection process. The basic edits are indicated by black polygons. The corrections are shown in red polygons.

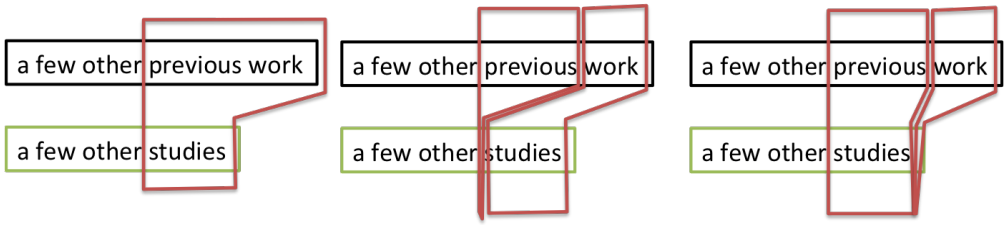
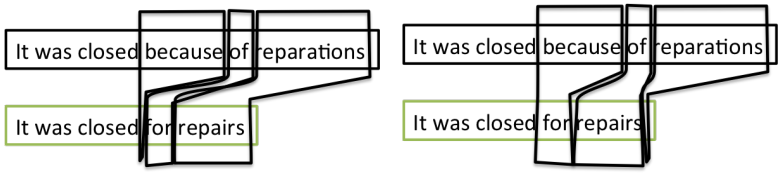
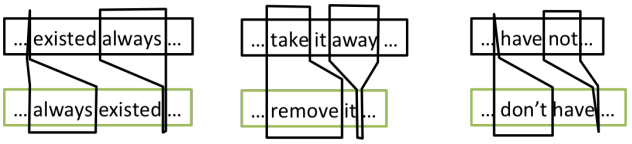


Figure 25: Ambiguity when interpreting the revision from "a few other previous work" to "a few other studies". I list three ways, where each red polygon indicates a separate correction. The first two are common interpretations of the revision. For humans, it is easy to tell that the third interpretation is less realistic. However, this is less obvious to computers.

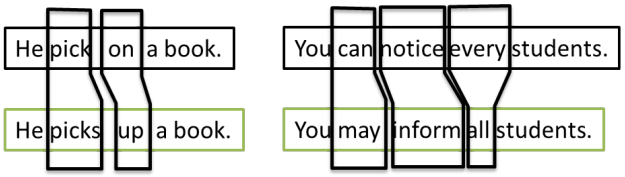


(a) (b)

Figure 26: Basic edits extracted by the edit-distance algorithm do not necessarily match our linguistic intuition. The ideal basic-edits are shown in Figure 26(a), but since the algorithm only cares about minimizing the number of edits, it may end up extracting basic-edits shown in Figure 26(b).



(a) The basic edits are addressing the same problem. But these basic edits are non-adjacent, and therefore not merged by S&Y's algorithm.



(b) The basic edits in the above two cases address different problems though they are adjacent. S&Y's merging algorithm incorrectly merges them.

Figure 27: Merging mistakes by the algorithm proposed in Swanson and Yamangil (2012) (S&Y), which merges adjacent basic edits.

wrong scope. These errors caused their system to mis-detect 30% of the corrections. Since mis-detected corrections cannot be analyzed down the pipeline, the correction detection component became the bottle-neck of their overall system. Out of the 42% corrections that are incorrectly analyzed[1], 30%/42%≈70% are caused by mis-detections in the first place. An improvement in correction detection may increase the system accuracy overall as well.

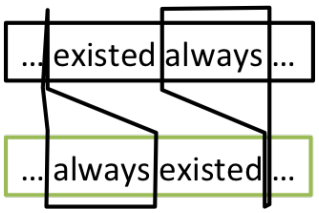
Between detecting basic-edits and merging basic-edits, which is accountable for more correction mis-detections? Which step requires more improvement? In Section 5.5.2 I conducted an error analysis under the same experimental setup as Swanson and Yamangil. I show that (1) the merging step accounts for most of the mis-detections; (2) around 75% correction mis-detections can be eliminated if we were able to employ an "oracle" merging algorithm which makes perfect merging decisions all the time. Therefore, to effectively reduce the algorithm's mis-detection errors, I propose to build a classifier to make more accurate merging decisions.

5.3 A CLASSIFIER FOR MERGING BASIC-EDITS

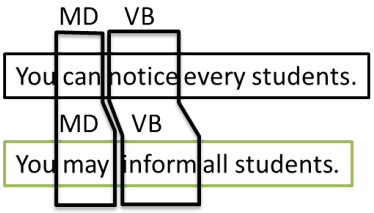
Figures 27 highlights the problems with indiscriminately merging basic-edits that are adjacent. Intuitively, it seems that the decision should be more context dependent. Certain patterns may indicate that two adjacent basic-edits are a part of the same correction while others may indicate that they each address a different problem. For example, as illustrated in Figure 28(a), when the insertion of one word is followed by the deletion of the same word, the insertion and deletion are likely addressing one single error. This is because these two edits would combine together as a word-order change. On the other hand, as illustrated in Figure 28(b), if one edit includes a substitution between words with the same POS's, then it is likely fixing a word choice error by itself. In this case, it should not be merged with other edits.

To predict whether two basic-edits address the same writing problem more discriminatively, I train a Maximum Entropy binary classifier based on features extracted from relevant contexts for the basic edits.

[1] Swanson and Yamangil reported an overall system with 58% F-score.



(a) The pattern indicates that the two edits address the same problem



(b) The pattern indicates that the two edits do not address the same problem

Figure 28: Patterns indicating whether two edits address the same writing mistake.

5.3.1 Features

I use features in Table 10 in the proposed classifier. I design the features to indicate: **(A)** whether merging the two basic-edits matches the pattern for a common correction. **(B)** whether one basic-edit addresses one single error.

5.3.2 Training

I train the classifier using samples extracted from revisions where individual corrections are explicitly annotated. I first extract the basic-edits that compose each correction. I then create a training instance for each pair of two consecutive basic edits: if two consecutive basic edits need to be merged, I will mark the outcome as *True*, otherwise it is *False*. We illustrate this procedure in Figure 29.

Type	name	description
A	gap-between-edits	Gap between the two edits. In particular, I use the number of words between the two edits' original words, as well as the revised words. Note that Swanson and Yamangil's approach is a special case that only considers if the basic-edits have zero gap both in the original and the revised sentences.
	tense-change	I detect patterns such as: if the original-revision pair matches the pattern "V-ing⇒to V".
	word-order-error	Whether the basic-edits' original word set and the revised word set are the same (one or zero).
	same-word-set	If the original sentence and the revised sentence have the same word set, then it's likely that all the edits are fixing the word order error.
	revised-to	The phrase comprised of the two revised words.
B	editdistance=1	If one basic-edit is a substitution, and the original/revised word only has 1 edit distance, it indicates that the basic-edit is fixing a misspelling error.
	not-in-dict	If the original word does not have a valid dictionary entry, then it indicates a misspelling error.
	word-choice	If the original and the revised words have the same POS, then it is likely fixing a word choice error.
	preposition-error	Whether the original and the revised words are both prepositions.

Table 10: My proposed classifier's features for merging decisions during correction detection.

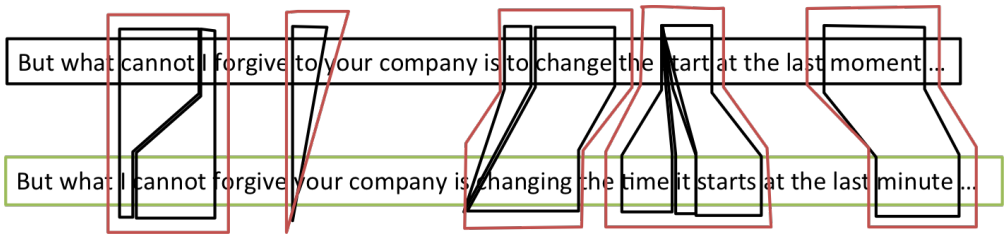


Figure 29: Extracting training instances for the merger. Our goal is to train classifiers to tell if two basic edits should be merged (*True* or *False*). I break each correction (outer polygons, also colored in red) in the training corpus into a set of basic edits (black polygons). I construct an instance for each consecutive pair of basic edits. If two basic edits were extracted from the same correction, I will mark the outcome as *True*, otherwise I will mark the outcome as *False*.

corpus	sentences	$\frac{\text{sentences with} \geq 2 \text{ corrections}}{\text{revised sentences}}$
FCE	33,900	53.45%
NUCLE	61,625	48.74%
UIUC	883	61.32%
HOO2011	966	42.05%

Table 11: Basic statistics of the corpora that I consider in correction detection.

5.4 EXPERIMENTAL SETUP

I experiment with different merging algorithms on their abilities to detect corrections from revisions.

5.4.1 Dataset

An ideal data resource would be a real-world collection of student essays and their revisions (Tajiri et al., 2012). However, existing revision corpora do not also have the more fine-grained annotations necessary for our experimental gold standard. I instead use error annotated data, in which the corrections were provided by human experts. I simulate the revisions by applying corrections onto the original sentence. The teachers' annotations are treated as gold standard for the detailed corrections.

I consider FCE, NUCLE, UIUC and HOO2011 corpora (Table 2), which all provide corrections along with error type mark-ups, but following different standards. The basic statistics of the corpora are shown in Table 11. In these corpora, around half of revised sentences contains multiple corrections.

I have split each corpus into 11 equal parts. One part is used as the development dataset; the rest are used for 10-fold cross validation.

Corpus	Error Types	Accuracy
FCE	73	80.02%
NUCLE	27	67.36%
UIUC	8	80.23%
HOO2011	38	64.88%

Table 12: Error type selection accuracies on different corpora. I use a Maximum Entropy classifier along with features suggested by Swanson and Yamangil for this task. The reported figures come from 10-fold cross validations on different corpora.

5.4.2 Evaluation Metrics

In addition to evaluating the merging algorithms on the stand-alone task of correction detection, I have also plugged in the merging algorithms into an end-to-end system in which every automatically detected correction is classified into an error type. I replicated the stand-alone error type selector described in Swanson and Yamangil (2012). The error type selector's accuracies are shown in Table 12[2] . I compare the following two merging algorithms:

S&Y The merging heuristics proposed by Swanson and Yamangil. We merge the adjacent basic edits into single corrections.

MaxEntMerger I use the Maximum Entropy classifier to predict whether we should merge the two edits, as described in Section 5.3.

I evaluate extrinsically the merging components' effect on overall system performance by calculate (1) the F_1-score in detecting corrections (2) the F_1-score in correctly detecting both the corrections' and the error types they address. I compute the number of corrections that have been correctly detected by comparing the actual corrections boundaries with the boundaries of our algorithm's detected corrections.

[2]My replication has a slightly lower error type selection accuracy on FCE (80.02%) than the figure reported by Swanson and Yamangil (82.5%). This small difference on error type selection does not affect my conclusions about correction detection.

5.5 EXPERIMENTS

I design the following experiments to answer two questions:

1. Do the additional contextual information about correction patterns help to guide the merging decision? How much does a classifier trained for this task help to improve the system's overall accuracy?

2. How well does my correction detection method generalize over different sources?

My major experimental results are presented in Table 13 and Table 15. Table 13 compares the overall educational system's accuracies with different merging algorithms. Table 15 shows the system's F_1 score when trained and tested on different corpora. I make the following observations:

First, Table 13 shows that by incorporating correction patterns into the merging algorithm, the errors in correction detection step were reduced. This led to a significant improvement on the overall system's F_1-score on all corpora. The improvement is most noticeable on FCE corpus, where the error in correction detection step was reduced by 9%. That is, one third of the correction mis-detections were eliminated. Table 14 shows that the number of merging errors are significantly reduced by the new merging algorithm. In particular, the number of false positives (system proposes merges when it should not) is significantly reduced.

Second, my proposed model is able to generalize over different corpora. As shown in Table 15. The models built on corpora can generally improve the correction detection accuracy[3]. Models built on the same corpus generally performs the best. Also, as suggested by the experimental result, among the four corpora, FCE corpus is a comparably good resource for training correction detection models with my current feature set. One reason is that FCE corpus has much more training instances, which benefits model training. I tried varying the training dataset size, and test it on different corpora. Figure 30 suggests that the model's accuracies increase with the training corpus size.

[3]I currently do not evaluate the end-to-end system over different corpora. This is because different corpora employ different error type categorization standards.

Method	Corpus	Correction Detection F_1	Overall F_1-score
S&Y	FCE	70.40%	57.10%
MaxEntMerger	FCE	**80.96%**	**66.36%**
S&Y	NUCLE	61.18%	39.32%
MaxEntMerger	NUCLE	**63.88%**	**41.00%**
S&Y	UIUC	76.57%	65.08%
MaxEntMerger	UIUC	**82.81%**	**70.55%**
S&Y	HOO2011	68.73%	50.95%
MaxEntMerger	HOO2011	**75.71%**	**56.14%**

Table 13: Extrinsic evaluation, where I plugged the two merging models into an end-to-end feedback detection system by Swanson and Yamangil.

Merging algorithm	TP	FP	FN	TN
S&Y	33.73%	13.46%	5.71%	47.10%
MaxEntMerger	36.04%	3.26%	3.41%	57.30%

Table 14: I evaluate the proposed merging model's prediction accuracy on FCE corpus. This table shows a breakdown of true-positives (TP), false-positives (FP), false-negatives (FN) and true-negatives (TN) for the system built on FCE corpus.

testing / training	FCE	NUCLE	UIUC	HOO2011
S&Y	70.44	61.18%	76.57%	68.73%
FCE	**80.96%**	61.26%	**83.07%**	75.43%
NUCLE	74.53%	**63.88%**	78.57%	74.73%
UIUC	77.25%	58.21%	82.81%	70.83%
HOO2011	71.94%	54.99%	71.19%	**75.71%**

Table 15: Correction detection experiments by building the model on one corpus, and applying it onto another. I evaluate the correction detection performance with F_1 score. When training and testing on the same corpus, I run a 10-fold cross validation.

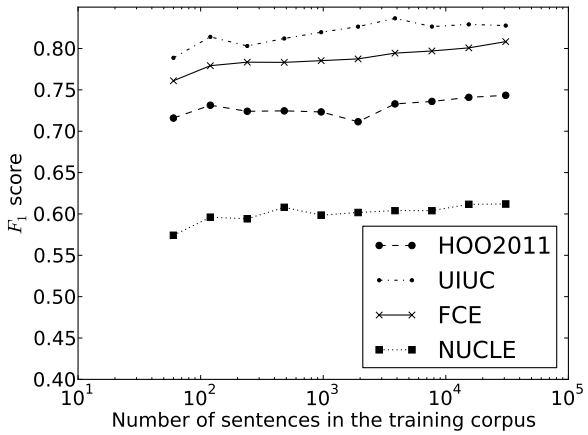


Figure 30: I illustrate the performance of correction detection systems trained on subsets of FCE corpus. Each curve in this figure represents the F_1-scores for correction detection of the model trained on a subset of FCE and tested on different corpora. When testing on FCE, I used $\frac{1}{11}$ of the FCE corpus, which I kept as development data.

training \ testing	FCE	NUCLE	UIUC	HOO2011
S&Y	76.57	79.01%	77.33%	80.50%
FCE	**86.95**%	75.72%	**83.83**%	86.52%
NUCLE	80.64%	**81.67**%	79.42%	86.63%
UIUC	83.04%	73.72%	83.56%	82.35%
HOO2011	77.28%	70.19%	71.77%	**87.37%**

Table 16: Correction detection experiments by building the model on one corpus, and applying it onto another. I use the same setup as in Table 15, except that I use a less restricted evaluation metric described in Section 5.5.1.

5.5.1 A Less Strict Evaluation

The current evaluation may under-estimates the algorithms' accuracies because it considers equivalent corrections to be different. Consider the last change in Figure 23, which involves changing "moment" into "minute". Some annotators may annotate this correction as a phrase rewrite instead: e.g. "last moment"⇒"last minute". In this case, even if the algorithm successfully detects the correction "moment"⇒"minute", my evaluation still considers it to be incorrect. To eliminate this effect, I relax the comparison such that two corrections are considered to be equivalent if applying the two corrections on the original sentence results in the same revision. The new evaluation result is summarized in Table 16. The result suggests that around 85% of the corrections can be successfully detected on most corpora when I build the merging model on FCE corpus. My proposed method still outperforms the baseline under the new evaluation.

5.5.2 Error Analysis for Correction Detection

When corrections are mis-detected, which step is responsible for it? Is it because my algorithm extracted basic-edits incorrectly? Or is it because my algorithm merged them incorrectly?

One way to know this is to try to examine the intermediate output from step one. If the inter-

method	dev corpus	errors caused by incorrect merges
S&Y	FCE	75.29%
S&Y	NUCLE	65.91%
S&Y	UIUC	91.67%
S&Y	HOO2011	93.33%
MaxEntMerger	FCE	63.40%

Table 17: Error break-down of correction detection models in development datasets.

mediate output is incorrect, then it is likely that the first step is responsible for the error. Otherwise, the error is caused by the second step. Ideally, we would like the first step to come up with basic-edits that have been performed by the language tutors. So, one idea of error analysis is to compare the first step's intermediate result with the actual basic-edits performed by the language tutor.

However, in practice, we often do not know what basic edits were performed. I propose to make an approximation instead. I consider the actual edits occurred within the sentence to be the set of edits that composed each correction that actually took place. I also considered that each correction was made with the smallest number of basic edits. These assumptions lead to approximating the actual basic-edits by extracting basic-edits for each correction using the Edit-Distance algorithm.

This approximation allows us to calculate the number of correction detection errors caused by incorrect basic edits, as well as those caused by incorrect merges. The latter number provides an approximation of the cases where a better merging algorithm could help reduce correction detection errors. Consider an "oracle" merging algorithm which takes in a set of basic edits, and is able to provide the right set of corrections whenever it is possible. The "oracle" would be able to obtain the correct set of corrections for the basic-edit sets we obtain with the approximation above. Therefore, if the first step ends-up with this set of basic-edits, an improved merging algorithm should be able to yield the right correction detection.

I conduct experiments on the development portions of the four corpora. As illustrated in Table 17, among sentences where the corrections are incorrectly detected, the basic edits are correctly

detected around 80% of the time on FCE, UIUC and HOO2011 corpora. This suggests that the merging phase accounts for around 80% errors. Although my proposed merging algorithm is able to significantly reduce correction detection errors, it still has space for improvement. A big portion of the correction detection errors can be reduced if we have an even better merging algorithm.

5.6 RELATED WORK

Correction detection involves extracting corresponding phrases between two sentences. To this end, phrase extraction (Koehn et al., 2003) and paraphrase extraction (Cohn et al., 2008) face similar challenges. Phrase extraction systems aim to improve the end-to-end MT or paraphrasing systems. A bigger concern is to guarantee the extracted phrase pairs are indeed translations or paraphrases. Recent work focuses on identifying the alignment/edits between two sentences (Snover et al., 2009; Heilman and Smith, 2010).

The fundamental difference is, however, the granularity of the extracted phrase pairs is a major concern in my work. This is because we need to guarantee each detected phrase pair to address exactly one writing problem. I conducted an error analysis in Section 5.5.2 which confirmed that deciding whether a phrase pair fixes one individual mistake is the bottleneck in the previous work (Swanson and Yamangil, 2012).

5.7 CHAPTER SUMMARY

A revision often contains multiple corrections that address different writing mistakes. I have investigated ways of accurately detecting individual corrections in one single revision. One major challenge in correction detection lies in determining whether consecutive basic-edits address the same mistake. I have shown that a classifier can be trained to for this task. My experiments suggest that: (1) the proposed classifier reduces correction mis-detections in previous systems by 1/3, leading to significant overall system performance. (2) my method is generalizable over different data collections.

6.0 CONFUSION SET CONSTRUCTION

6.1 CHAPTER OVERVIEW

Constructing automated systems that provide feedback to ESL learners require a great deal of human involvement. This chapter focuses on building one important Grammatical Error Correction (GEC) system component, the confusion sets. One word's confusion set is composed of a list of words that learners are most likely confused between. Previously confusion sets were built with extensive help from language experts, either to manually filter the initially large confusion sets (Liu et al., 2010), or to annotate large ESL corpora (Rozovskaya and Roth, 2010c). I hypothesize that such involvement is not necessary. Intuitively, confusions between word usages were developed while learning English words. I propose to construct confusion sets using models that simulate word learning.

Constructing confusion sets requires developing models to capture the key factors of word learning. Previous researches mainly consider two aspects of word learning: their out-of-context meaning and in-context usage. The out-of-context meaning is often implied by its definition and its relations to other words. The in-context usages is characterized by the circumstances to use a word. They can be captured by semantic similarity models and language models, respectively.

In this chapter, I investigate the following questions:

1. Can we build models to learn about word confusions, without human labeling?
2. Can we build a model that incorporates both the learning of out-of-context meanings and in-context usages?

I have conducted a study that explores simulation models to capture the confusions between words. My studies in particular focused on building confusion sets for prepositions. I propose to

use RCA model, which can simulate the learning of words' out-of-context meanings and in-context usages. I compare the proposed model against two models that simulate how learners obtain words' out-of-context meaning and in-context usage separately. Experimental results suggest that:

1. by considering the interaction of out-of-context meaning and in-context usages, my proposed model produces better confusion sets than those which consider them separately.
2. the resulting confusion sets are competitive with those directly learned from an error-annotated ESL corpus containing 150K preposition usages.

6.2 CONFUSION SETS

Confusion sets play an important role in GEC systems that correct single word usage errors. Generally speaking, reducing the confusion set helps lead the classifiers in the GEC system to a better performance by prohibiting them from considering the outcomes that are both *unlikely* and *misleading*. For example, although ESL learners normally would not confuse *within* with *in*, classifiers may have difficulties telling them apart. Therefore, eliminating *within* from *in*'s confusion set may help the classifier. Generally speaking, by reducing the confusion set's size to rule out these outcomes, although the systems will be disabled from correcting certain types of mistakes, they will often increase the accuracies on more prevalent error types and finally lead to a better overall performance. In the past, Rozovskaya and Roth (2010c) showed that by limiting the size of the confusion set for prepositions, their GEC system's performance improved.

Currently, confusion sets were built with language tutors' manual efforts. The major challenge in automatically building confusion sets is developing a model that captures the most relevant factors for confusions. Many factors may contribute to confusion between word usages. For instance, Dahlmeier and Ng (2011a) observed that ESL collocation errors may be due to similarities between two words' spellings, pronunciations, synonyms, and paraphrases in the writer's native language (L1). However, by including all words that are similar according to any of these factors, one would end up with a large confusion set which causes difficulties for the classification tasks down the GEC pipeline.

Confusion between words mainly result from their similarities during learning. Therefore,

picking the model to simulate learning is a fundamental decision. Computational Linguistics literature provides two views on how to build models for learning words. One view is that learning words means understanding the words' meanings and relations to one another. I refer to this view as capturing the words' *out-of-context meanings*. Another view is that learning words means understanding which word to choose under which conditions. I refer to this view as capturing the words' *in-context usages*.

In *lexical semantics*, people hold the first view. In this area, researchers try to find how and what words mean, denote, and their relations/similarities . This view tends to explain the cause of confusions to the similarities between words. They build confusion sets containing words that are similar in semantic meanings.

In *language modeling*, people hold the second view. People consider the ability of choosing the appropriate word under each context to imply the mastery of the *language*, which includes the understandings of the *words* in the language. This view tends to explain the cause of confusions to the learners' incapability to completely manage how to use words.

I believe the knowledge of word out-of-context meanings and in-context usages build on top of each other while learners learn English words. Therefore I propose a model that reflects the interactions between the understandings of words' out-of-context meanings and in-context usages. I show via empirical studies that the proposed model builds confusion sets that both improve GEC systems' performance, and correlate well with real ESL mistakes.

6.3 AUTOMATIC CONFUSION SETS CONSTRUCTION

ESL writers are more likely to confuse words that they find to be similar during their language learning. In this section I present three models that simulates how ESL learners might learn words. In the first two subsections, I describe models of separately learning words' out-of-context meanings and in-context usages, respectively. In the last subsection, I introduce a model that is optimized for learning the out-of-context meanings and in-context usages of words all together. Within each subsection, I also develop the reason of ESL writers' confusions, and propose the corresponding way to automatically construct confusion sets.

6.3.1 Learning Words' Out-Of-Context Meanings – Distributional Models

Under an *out-of-context meaning based* perspective, a learner's primary goal is to understand word meanings, and it is the similarities between words' out-of-context meanings that cause word choice confusions. However, this is not to say that learners ignore word usages. Indeed, although dictionary entries contain direct definitions of words, researches in the past showed that learners do not learn by memorizing dictionary entries; instead, they infer words' meaning/function from the context, and then connecting the new words to the words they are already feel familiar (Fischer, 1990). Under this perspective, learning the in-context usages of words is not explicit, it is a means to achieve the primary goal of understanding word meanings.

To simulates an out-of-context meaning based learner, I build a model of word similarity metrics from processing standard English text. Specifically, I build *distributional models* in which the similarities of words are calculated from a comparison of the contexts they appear in (Pereira et al., 1993; Lin, 1998; Lee, 1999). Then, to fill in a word's confusion set, I pick the words that are most similar according to the metric. Pantel and Lin (2002) showed this method is able to yield similarities that correlate well with the similarities of words' out-of-context meanings.

In my work, I calculate the words' out-of-context meaning similarity by using a distributional model (Pereira et al., 1993; Lee, 1999), in which each preposition is represented as a distributional vector of its context features. Examples of usage contexts that have been shown to be relevant for the task of preposition selection in previous work (De Felice, 2008; Tetreault et al., 2010a; Dahlmeier and Ng, 2011a) include:

Gov: the syntactic dependency governors of the preposition

Obj: the dependency objects of the preposition

GovTag, ObjTag: the part-of-speech tags of the dependency governors and objects

L1-Trans: L1 translations of the preposition

I employ **Gov, Obj, GovTag, ObjTag** features to capture the grammatical context of the preposition selection. I also employ **L1-Trans** to capture both the intended semantic meaning of the preposition and the **L1** background information which was shown to be relevant to confusions (Rozovskaya and Roth, 2010c; Dahlmeier and Ng, 2011a).

The distribution of each preposition's usage context can be estimated from a standard English corpus. Then the similarity between any pair of preposition vectors can be computed using common distance metrics such as: KL-Divergence, Euclidean distance, and cosine similarity.

This approach, however, may not be appropriate for our problem for the following two reasons:

Firstly, under a distributional model, two prepositions are considered similar only if the distribution of all their usages are similar. This is a strong restriction in the sense that two prepositions might only be similar under certain specific usage contexts but are not generally similar. For example, the prepositions *of* and *for* typically have fairly distinctive usages; however, ESL writers often confuse the two if the previous word was *need*.

Secondly, even if two words have similar usages under certain usage context, i.e. have similar probabilities of being used(e.g. both with 0.2 probability), people still may not be likely to confuse them with each other – instead, they are more likely to confuse them with a third word which have higher probabilities(e.g. 0.5). This is because the learner is more likely to pick the word that seems most plausible in the context, if without further information.

6.3.2 Learning Words' In-Context Usages – Preposition Selector

Under an *in-context usage-only* model, it is assumed that the learners' main goal is to understand how to choose words in a given context, and that they learn about such knowledge from standard English text. Because classifiers can also be trained to choose words, I simulate ESL learners' learning process as training a classifier for the word selections task (Tetreault and Chodorow, 2008; Tetreault et al., 2010a) on standard English text. The trained classifier can be seen as a type of language model: given a context, it predicts the most likely word in that context.

Under this model, it is expected that the word choice confusions are mainly caused by the learners' incapability to completely master the word usages. Therefore, to see what confusions an ESL learner may have, I then rerun the trained classifier on the training data to collect the mistakes it makes.

6.3.3 Learning Both Out-Of-Context Meanings and In-Context Usages – RCA

I believe the knowledge of word out-of-context meanings and in-context usages build on top of each other while learners learn English words. Therefore I propose a model that reflects the interactions between the understandings of words' out-of-context meanings and in-context usages; it works toward making the out-of-context meanings and in-context usages compatible with each other. Similar with the model in section 6.3.1, in the end, I build words' confusion sets by filling them in with words that are most similar in their out-of-context meanings.

My new model simulates that, while reading English texts, learners adjust their understandings if one word's out-of-context meanings mismatches with what the context suggests. To illustrate this idea, suppose that initially a learner thinks: (1) *for* means "on someone's behalf" (2) *to* means "the target is". When he/she reads "The gift is *for* you", he/she will notice a mismatch between "for" and its context. To the learner, "to" may seem more appropriate than "for" in this context. This is because "is __ you" indicates that the word in between suggests a target, which matches with his/her understanding of "to". To overcome the mismatch, the learner will adjust his/her understandings about the meanings of "for" and "to" accordingly. Later, he/she may think that, say: *for*'s meaning includes "the target is"; while *to* means "the target is" only in certain contexts.

My model tries to simulate this process. I illustrate the idea in Figure 31. In particular, my model represents learners understandings about meanings as points in an Euclidean space, and uses distances between them to measure the mismatches. In Figure 31, The black points represent the meaning of words themselves; the green points are meanings suggested by occurrence contexts. When a word occurs within a certain context, I will draw a red line between the corresponding two points, indicating the mismatch between the word and its context. The learner's learning goal is to reduce the mismatches, by adjusting his/her understandings. In our illustration, this would involve moving the green and black points to reduce the red lines' lengths. Note that one trivial solution to minimizing the lengths would be assigning all points to one single point. But we should avoid this trivial solution, since no two words are identical. We achieve this by enforcing non-zero distances between preposition's points.

I formalize this learning process mathematically to help with our simulations.

MEANING SPACE I assume that all possible meanings may be embedded in an Euclidean space

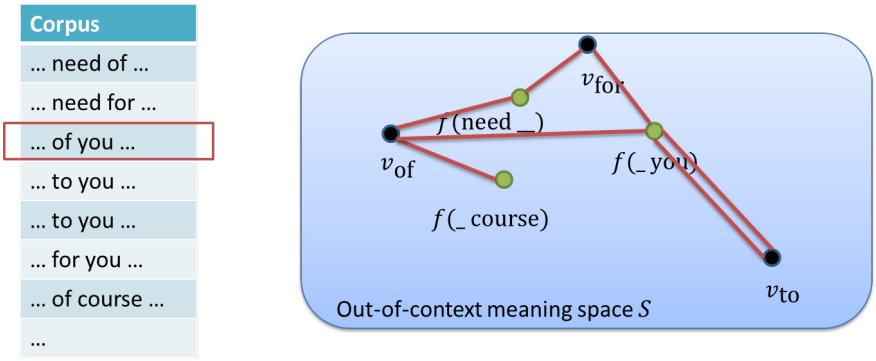


Figure 31: Mathematical formulation of ESL learners' word learning process in my proposed model. I use green points to represent the meanings suggested by a word's occurrence context, and black points to represent the meaning that the word itself suggests. The farther the two points reside, the bigger the difference. For each word occurrence, I draw a red line between the word and its context, representing the mismatch between them. By adjusting the green and black points, we may minimize the overall mismatches.

S. Because I mainly focus on n prepositions $w_1, \ldots, w_n$, I assume that all out-of-context meanings during learners' learning process can be described by a linear interpolation of the n prepositions' meaning vectors. That is, the subspace containing all out-of-context meanings learners consider has at most n dimensions. I therefore may assume $S = \mathbb{R}^n$, without loss of generality.

PREPOSITIONS' MEANINGS The n prepositions $w_1, \ldots, w_n$ have corresponding out-of-context meanings $\vec{v}_1, \ldots, \vec{v}_n \in S$. Further, I denote $V = (\vec{v}_1, \ldots, \vec{v}_n)$, $I_i = (\underbrace{0, \ldots, 0}_{i-1}, 1, \underbrace{0, \ldots, 0}_{n-i})^T$, so that $\vec{v}_i = VI_i$. I ensure that these points do not clutter by forcing $|\det(V)| \geq 1$[1].

OUT-OF-CONTEXT MEANING DECISIONS I model the learners understanding about the meaning suggested by the context as a function $\vec{f}$ which maps a context C to a point in the meaning space $\vec{f}(C) \in S$. This points to the out-of-context meaning the learner would like to choose under context C. C is in the format of a set of relevant contextual features for the preposition choice decisions. Following the discussion in section 6.3.1, I consider the relevant contextual features

[1]Because $|\det(V)|$ is the area circled by the word vectors in V, forcing it to be higher than 1 can be interpreted as assuming the learners know beforehand that the prepositions cannot be too similar to the others.

Gov, Obj, GovTag, ObjTag, L1-Trans.

MISMATCHES For one word usage sample (C, w_i), where C is the context, and w_i is the actual preposition choice, I define the mismatch of the word choice by $||\vec{f}(C) - \vec{v}_i||^2$. This means: the more difference between the learner's expected word choice $\vec{f}(C)$ and the actual word choice $\vec{v}_i$, the bigger mismatch the learners finds.

LEARNING GOAL I assume that the learners learn about the word usages from some standard English corpus[2] D, containing word usage samples in the format (C, w_i). The learners' learning goal is to find V and $\vec{f}$ which get them most "comfortable" with the word usages in D. Therefore, mathematically, the learners' objective is to minimize the mismatches on the English text D: $\min_{\vec{f},V} \sum_{(C,w_i)\in D} ||\vec{f}(C) - v_i||^2$.

Following the discussion above, I formalize the learning process of ESL learners as finding the best *uncluttered* word vectors V and word usage patterns $\vec{f}$ which together minimize the mismatch function over some standard English corpus D:

$$\min_{\vec{f},V} \sum_{(C,w_i)\in D} ||\vec{f}(C) - VI_i||^2 \quad \textit{s.t.} |\det(V)| \geq 1 \tag{6.1}$$

I calculate the optimal set of word vectors V in the optimization problem above by firstly reducing the problem into a Minimization of Within Class Distances problem, as shown in Appendix A.1, and then solving it using the Relevance Component Analysis(RCA) algorithm (Bar-Hillel et al., 2006).

In the end, I will be able to obtain the word vectors for prepositions $VI_1, \ldots, VI_n$, and therefore also their similarities by calculating the distance between them (the distance between prepositions w_i, w_j is $||VI_i - VI_j||$). According to my model's assumption, after ESL learners' learning, the similarities of out-of-context meanings of two words' will highly correlate to this distance. I can therefore fill in the confusion set for every preposition with the prepositions that have the least distances to it.

This approach is similar to the approach described in Section 6.3.2 in that it also focuses on the similarities of preposition usages under specific contexts. The two approaches differ, however, in their treatments of the degrees to which words are considered to be similar. For example, consider

[2] Although there may be other sources where the learners may obtain English knowledge from, such as dictionaries, the learners would learn word usages better from texts (Fischer, 1990).

a corpus where under some certain context C, prepositions p_a, p_b and p_c occur 101, 100, 100 times, respectively. Using RCA, the system would consider all three to be mutually confusable because they appear almost equally frequently in the same context. On the other hand, while the preposition selector considers p_b and p_c to be confusable with p_a, it does not conclude that p_b and p_c are also mutually confusable under context C.

Thus, if most usage contexts contain only one or two preposition types, the preposition selector and RCA may produce similar confusion sets; but if the data also include usage contexts that contain three or more preposition types, RCA may offer confusion sets based on a more globally optimized similarity metric.

6.4 EXPERIMENTAL SETUP

I conduct experiments to compare different methods for constructing confusion sets. To evaluate the confusion sets' qualities, I examine how they impact the performance of an end-to-end grammar error correction (GEC) system. In particular, I train a separate classifier for each preposition using only training examples that are covered by the confusion set, a setup similar to the **NegL1** system as described in (Rozovskaya and Roth, 2010c). Additionally, I also compare the confusion sets with an intrinsic evaluation; I measure how well each method's confusion sets match real ESL mistakes by calculating their *coverage* on an annotated ESL corpus.

6.4.1 Data

I use NUCLE as the ground-truth for my experiments. In this collection, many writers' native (L1) language is Chinese. Following the methodologies established in other studies on the preposition selection problem, I focus on the 36 most frequent prepositions[3]. I used 80% of the full corpus for training, 10% for development and 10% for testing.

I use the NUCLE corpus in several ways. First, it is used to establish upper-bound confusion

[3]These preposition words include *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*

sets. I constructed these "gold" confusion sets by tabulating the observed preposition errors in the corpus. Second, it is used as a source of training data for the end-to-end GEC system[4]. For each confusion set construction method, I extract from the training portion of NUCLE those instances that are consistent with the proposed confusion sets to train the GEC system. The trained systems are then tested on the unfiltered test set. Third, it is used as the ground truth for computing the coverage metric.

The non-ESL corpus used for constructing confusion sets is the Foreign Broadcast Information Service (FBIS) corpus, which is a Chinese-English bilingual corpus. For most experiments, only the English portion is used. For experiments that make use of L1 translations, I extracted the Chinese translations for English prepositions using the GIZA++ (Och and Ney, 2004) implementation of the IBM word alignment model (Brown et al., 1993). Of the FBIS corpus, I used its first 32,000 sentences, which contain 151,767 prepositions.

6.4.2 Metrics

6.4.2.1 Extrinsic Evaluation I use F_1-measure to evaluate the confusion sets' effects on the GEC system.

The GEC system is trained on a subset of NUCLE to ensure an above-zero error detection rate. Directly using NUCLE as training data will result in a "silent" system that prefers not to detect any errors. This is because it will adapt to the low error rate in NUCLE, where only 1.3% of the preposition instances contain an error, as learners do not make mistakes on most of the usual cases. To reduce the class imbalance for the underlying classifiers during training, I follow the methodology used by Dahlmeier and Ng (2011b) to keep all instances that contain an error and retain a random sample of q percent of the correct instances in the training data. In my experiments, the value of q ($20\% \leq q \leq 40\%$) is tuned on development data. I keep the test data as it is. That is, the filtering I discussed above is only applied on the training data.

[4]While using NUCLE to train the GEC system seems in contradiction with my overall aim of reducing my reliance on error-annotated corpus, I argue that the usage is appropriate here because I need to compare different approaches of constructing confusion sets without interference from other factors. I do not pursue alternatives such as injecting noise into standard English as training data (Rozovskaya and Roth, 2010c,a) to avoid unintended interactions between the confusion sets and the error generation methods.

6.4.2.2 Intrinsic Evaluation: Coverage

When an ESL student mistakenly uses some preposition instead of the correct one, the wrong preposition is not necessarily in the proposed confusion set list. I refer to the proportion of ESL students' mistakes in a corpus that fall into the proposed confusion set list as the *coverage* of the confusion set list on that corpus.

The coverage metric can be seen as measuring *recall*: how well does the proposed confusion set table cover the mistakes in some ESL corpus? If each confusion set includes all the prepositions, then the coverage would be 100%. As discussed earlier, in order for the confusion sets to be useful, they cannot be too large. A high quality confusion set table is one whose confusion sets are small in their sizes but cover the majority of the mistakes seen in the ESL corpus.

6.4.3 Confusion Set Construction Methods

My experiments compare the following confusion set construction methods:

THE TRIVIAL CONFUSION SETS(ALL PREPS) To show the confusion sets' effect in general from comparison, I establish a baseline by using the trivial confusion sets, in which all prepositions are considered to be confusable to each other.

CONSTRUCTION FROM NUCLE(GOLD) I establish the upper-bound of the confusion set table by tabulating the preposition mistakes in NUCLE. This confusion set table contains the most prepositions, and therefore is the one with the highest coverage of ESL mistakes.

CONSTRUCTION BY DISTRIBUTIONAL SIMILARITY METRICS As described in Section 6.3.1, this model represents a preposition as a feature vector and directly computes the distance between pairs of prepositions to construct confusion sets. The values of the feature vectors are computed from the FBIS corpus. Three standard distance/similarity measures are used: KL-Divergence(kl div), Euclidean Distance(euc dist) and Cosine Similarity(cos sim).

CONSTRUCTION FROM PREPOSITION SELECTOR ERRORS(SELECTOR) Section 6.3.2 proposes generating confusion sets from classification errors. Here, I train a Maximum Entropy classifier for the preposition selection task on the FBIS corpus, and rerun the classifier on the same data to collect the mistakes it still makes.

CONSTRUCTION BY WORD USAGE SIMILARITY MODELING(RCA) In Section 6.3.3 I proposed to simulate ESL learners' learning of both words' out-of-context meanings and in-context

usages. I formalize their learning as an optimization problem and then calculate words' out-of-context meanings and in-context usages using the RCA algorithm (Bar-Hillel et al., 2006). The final confusion sets contain words which have similar out-of-context meanings.

6.4.3.1 Fixing Sizes of Confusion Sets My evaluation fixes the size of the confusion sets in the final confusion set tables to be N, where $3 \leq N \leq 7$. This is mainly because confusion sets tables with sizes greater than 7 are able to cover over 90% of the ESL mistakes, and increasing confusion sets' sizes from there start to hurt the GEC systems' performance. On the other hand, when the sizes are too small, the confusion set lists prevents the GEC system from making reasonable corrections.

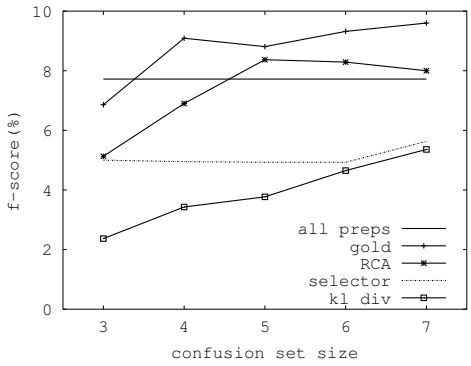
6.5 EXPERIMENTS

I compare the proposed methods of constructing confusion sets by using the resulting confusion sets in an end-to-end GEC system as described in Section 6.4. The experiments aim to address the following questions: (1) How does the proposed method for automatically constructing confusion sets from non-ESL corpus compare against those developed from error-annotated ESL corpus? (2) Regarding the models for ESL learners' word learning, does considering the interactions between their learning of words' out-of-context meanings and in-context usages help to capture the learners' confusions? (3) How are these models affected by the choices of different context feature groups? (4) How would the quality be affected by the choice of confusion set sizes?

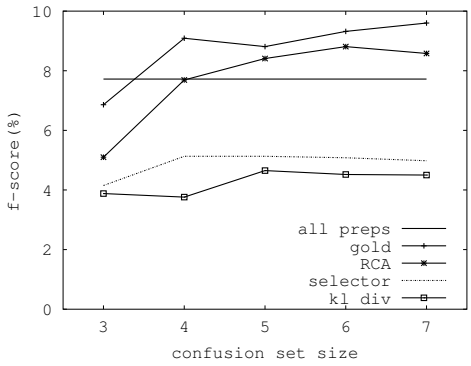
Figure 32 shows a summary of the results. Each plot shows the GEC system's *performance* versus the *size* of the confusion sets for each confusion sets construction method under a different set of context feature choices [5]. In the baseline all preps, because I always fix the confusion sets' size to be a constant number 36 to contain all prepositions, the resulting curves are displayed as horizontal lines in the figures.

I make four observations:

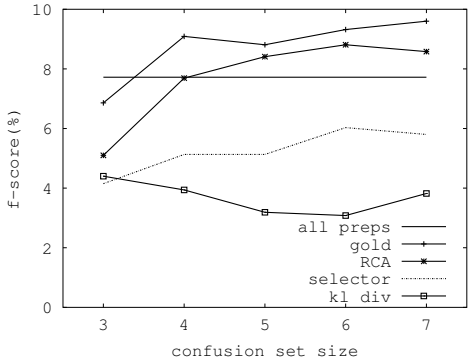
[5]Note that among the standard similarity metrics, I only plot kl div's F_1-scores because it performs better or similar to the other two methods in most of the cases. In later experiments, I will also only demonstrate the best of the three when all of them are performing similarly.



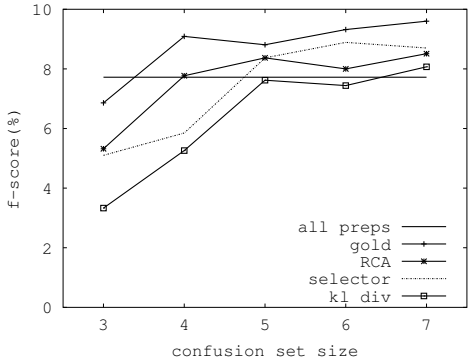
(a) Using **L1-Trans**



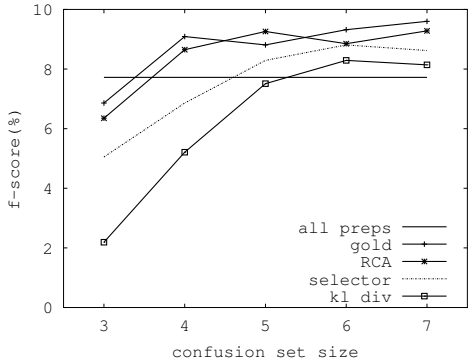
(b) Using **GovTag,ObjTag**



(c) Using **GovTag,ObjTag,L1-Trans**



(d) Using **Gov,Obj**



(e) Using **Gov,Obj,L1-Trans**

Figure 32: F_1-Scores of different confusion set construction methods. For each of the five feature combinations, a plot demonstrates the performance of different methods using that feature combination. I display every method's performance as a curve in which each point represents the GEC system's F_1-Score when using that method to construct confusion set list of a particular size for the 36 prepositions.

First, regarding the use of non-ESL corpus, the experimental results suggest that confusion sets that are automatically constructed from non-ESL corpus is competitive with those constructed from an error-corrected ESL corpus. When picking the best feature sets **Gov,Obj,L1-Trans** in RCA, the GEC system can perform as well as if it were using the gold confusion sets constructed from a corpus containing 150K preposition usages.

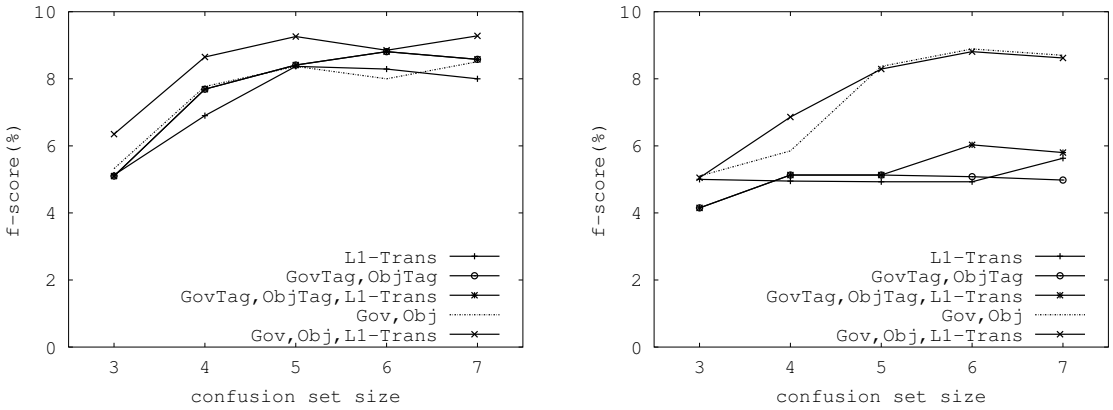
Second, regarding the models for ESL learners' word learning, my experiments suggest that the learners' confusions are better captured when I model their learning of both words' out-of-context meanings and in-context usages altogether. In my experimental results, confusion sets constructed by RCA model, which considers the interaction of words' out-of-context meanings and in-context usages, consistently outperforms the other automatic methods selector, kl div, euc dist, cos sim, which only consider the learning of either words' out-of-context meanings or their in-context usages.

Third, regarding the feature sets used in constructing confusion sets, I find that in general all the models tend to perform better when they use more features. For example, by using **Gov,Obj** in addition to **L1-Trans**, selector raises the GEC system's F-score from 5.00% to 8.81%. RCA, however, is more stable with respect to the feature set changes. I separately show, for these two models, a comparison of the features' effects on them in Figure 33.

Fourth, my evaluation confirms that, in general, using confusion sets helps improving the GEC system's performance. This is because by limiting the confusion set's sizes, one can greatly reduce the underlying classifiers' mis-classification errors, at the cost of reducing their coverage a little. These two factors together lead to positive changes overall. To further demonstrate this effect, I show in Table 18 statistics of the decomposition of GEC systems' errors on the testing dataset. Also worth noticing is that my proposed approach (RCA), although having a slightly less coverage compared to the (gold), reduces mis-classification errors even further.

6.5.1 Discussions

The experiments above demonstrated RCA's strength over other methods. In this section I provide more in-depth analysis on the differences between RCA and other methods, by comparing those methods' effects on the GEC system's precision and recall separately.



(a) RCA methods' F_1 scores

(b) Selector methods' F_1 scores

Figure 33: F_1-scores of models using different feature sets to build confusion sets for all 36 prepositions.

I fix the feature set that all methods use to be **Gov,Obj,L1-Trans** in the discussion, because it allows all models to perform their best.

6.5.1.1 Precision In Figure 34, comparing with the all prep baseline, we see that by limiting the classifiers' choices, confusion sets are indeed able to raise up GEC systems' precision. The confusion sets computed by RCA and euc dist are more helpful in raising the GEC system's precision, in contrast with selector. The difference is more significant when the confusion sets are small.

6.5.1.2 Confusion Set Coverage Furthermore, I would like to provide an analysis of the GEC system's recalls, which is, in my setup, mainly *affected* by the number of ESL mistakes that are precluded from classifiers' consideration by the confusion sets. I measure this by calculating the proportion of ESL mistakes they cover using the metrics developed in 6.4.2.2. The coverage also reflects one confusion set's match to ESL students' real mistakes.

Shown in Figure 35 are the coverage of confusion sets constructed by different models, of different sizes. RCA and the selector greatly outperform other automatic approaches.

	all	size=3		size=4		size=5		size=6		size=7	
	preps	gold	RCA	gold	RCA	gold	RCA	gold	RCA	gold	RCA
Out of Coverage	0	54	58	37	42	33	37	24	29	22	19
Mis-classification	284	135	119	162	147	173	158	189	176	203	195

Table 18: Confusion sets help reducing mis-classification errors. Here I categorize the GEC system's mistakes by whether they are caused by the confusion sets. *Out of Coverage* represents the cases where confusion sets precluded the right correction to be made, while *Mis-classification* includes all the other cases where the underlying classifiers are responsible for the prediction mistakes. The RCA I demonstrate here uses **Gov,Obj,L1-Trans** features.

6.6 RELATED WORK

Currently, confusion sets are primarily built in two ways that both consume language tutors' efforts. One way is to ask human experts using their knowledge about ESL mistakes to restrict the confusion set. This is the approach taken by Liu et al. (2010) for their GEC system for verb selection. Another alternative is to make use of an ESL corpus in which the mistakes have been corrected by an English teacher; in this case, the confusion sets can be tabulated from the annotations (Rozovskaya and Roth, 2010c; Dahlmeier and Ng, 2011a). A benefit of the corpus-driven approach is that the resulting confusion sets provide a reliable estimation of the distributions of the underlying error patterns. However, this type of annotated corpora takes time and effort to develop. Moreover, even when an ESL student makes many mistakes, the proportion of the writing that contains no error is still much greater. For example, in the NUS Corpus of Learner English (NUCLE) corpus (Dahlmeier and Ng, 2011b), there are a total of 3,302 preposition mistakes out of a total of 147,087 prepositions. Therefore, to build confusion sets for open class words such as verbs, one would need a very large annotated corpus.

Recent work automatically fill confusion sets with words that are similar in certain aspects. Dahlmeier and Ng (2011a) filled in each word's confusion sets with words that have similar spellings, pronunciations, along with their synonyms, and paraphrases in the writer's native language (L1). Liu et al. (2010) filled in word's confusion sets with their synonyms and words that

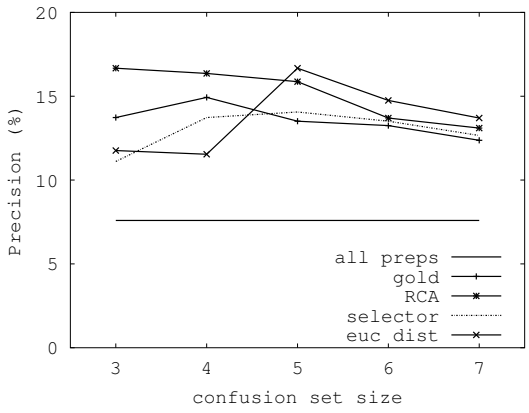


Figure 34: Precision when using confusion sets generated by different methods on NUCLE.

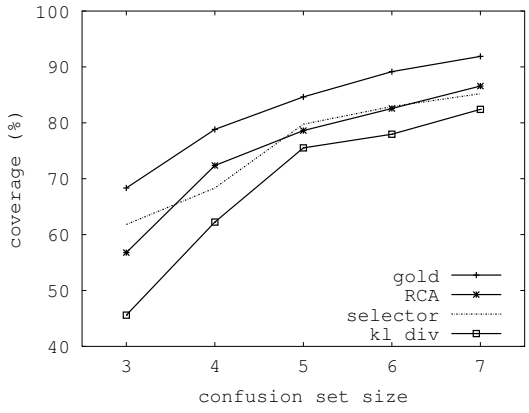


Figure 35: Coverage of confusion sets in different models using features **Gov,Obj,L1-Trans**

have similar SMT systems' phrase table entries (Och and Ney, 2004). Less effort was spent on limiting the sizes of the resulting confusion sets. As a result, the confusion sets may contain many words that pose difficulties to the classifier down the GEC pipeline.

My work further quantifies the similarities between words and their confusion set entries with the help of standard English corpora. The similarity measure helps us to limit the confusion set size, while not losing much coverage.

6.7 CHAPTER SUMMARY

I have investigated a method for automatically constructing confusion sets for preposition errors. It does not rely on any annotated ESL corpus or human post-processing. Based on the notion that ESL word selection errors are mainly because ESL learners are not able to choose between similar words, I have developed a model to analyze which words might seem similar to each other to an ESL learner. My model applies an algorithm called Relevance Component Analysis (Bar-Hillel et al., 2006) to describe how an ESL learner might learn both words' out-of-context meanings and in-context usages from reading English text. The resulting confusion sets have been shown to both improve GEC system's performance, and correlate well with real ESL mistakes. Also, by modeling the interaction between the out-of-context and in-context knowledge in ESL learners' learning, my model ends up with better confusion sets than the models considering the development of *only* out-of-context or in-context knowledge. One key strength of my proposed technique is that because it only relies on standard English corpora, it is more scalable.

7.0 CONCLUSIONS

I have investigated the feasibility of easing language tutors' workload through better computer support using NLP and machine learning methods. I have shown that with the appropriate underlying computational models, we can develop programs that reduce human efforts in detecting and correcting ESL learners' errors. In particular, in this dissertation work, I have developed computational models for the following problems:

Detecting local redundancies I have devised a metric for quantifying a word's potential of being redundant. By including this metric as a feature, the overall accuracy of the automatic redundant word detector nearly doubled, and it is seven times better than the random baseline. I have also developed a classifier for determining whether a sentence contains any redundant word with a 70% accuracy. Although the absolute accuracy is not yet high enough for downstream applications, my work suggests that local redundancies do indeed share some regular patterns, and points to a promising direction for a fully automated local redundancy detection system.

Detecting correction rationales I have developed an improved method for detecting the reasons for individual corrections between an original draft and its revision. I have shown that a classifier can be trained to better determine whether consecutive basic-edits address the same mistake. The proposed classifier reduces the mis-detections in the best reported previous system by one third.

Building confusion sets I have investigated automatically constructing confusion set by simulating learning both words' out-of-context meanings and in-context usages, using RCA algorithm (Bar-Hillel et al., 2006). The resulting confusion sets have been shown to both improve GEC system's performance, and correlate well with real ESL mistakes. Adapting the model reduces

tutors' involvement in building GEC systems.

In addition to the above enhancements, I enabled the following resources:

1. A framework for detecting redundant word.
2. Corpus annotations for word/sentence level redundancies.
3. An open-source system for identifying correction rationales between revisions.

FUTURE WORK

Improving the coverage and accuracy of computer systems to support second language learning is an important area of research. My dissertation work points to several directions for further research.

Better Local Redundancy Annotations As part of the redundancy detection research, I have developed a small corpus based on a subset of NUCLE corpus that contain clear cases of redundancy judgments. The corpus could be expanded both in terms of its size and its coverage. In particular, it ought to support a more diverse set of both the redundant cases and the non-redundant cases. Researchers may also benefit from a clearly defined guideline that differentiate between categories of redundant phrases.

Extend to to a Larger Text Unit All the works described in this dissertation have focused on writing issues within single sentences. But many writing issues also occur across sentences. Take redundancy detection as example: an entire sentences, or even an entire paragraph could be redundant within an essay. This happens when the learner states the same idea multiple times or writes hollow sentences. Another example is identifying correction rationales for cross-sentence edits. Suppose sentences in a paragraph are reordered; can computers determine the reason for these edits? These issues are more challenging because they are concerned with not only grammar sophistication but writing skill as a whole. Therefore, they are prevalent not only in ESL writings, but also in native learners' writings.

Building Confusion Sets for Larger Word Classes My research on confusion set examined the case for prepositions, one of the most common grammar mistakes. We may also extend the same framework to other word classes, such as verbs and nouns. However, learning confusion sets for larger word classes may require the classifier to capture different patterns. For example,

learners' native language may play a more important role in learning verbs/nouns, compared to prepositions. As a result, we may need different feature sets for these word classes.

With further research into new models of ESL errors, GEC systems will continue to improve and provide invaluable assistance to both ESL students and their tutors

APPENDIX

A.1 SOLVING THE OPTIMIZATION PROBLEM FOR WORD USAGE SIMILARITY

To solve the minimization problem in formula 6.1, we will first cast it into a Minimization of Within Class Distances problem.

Firstly, suppose there are N unique contexts $C_1, \ldots, C_N$ in the corpus, note that by grouping the samples with same contexts together, we may rewrite formula 6.1 as:

$$\min_{f,V} \sum_{1 \leq k \leq N} \sum_{(C_k, w_i) \in D_k} ||\vec{f}(C_k) - VI_i||^2 \quad \textit{s.t.} |\det(V)| \geq 1$$

where $D_k = \{(C, w_i) \in D \mid C = C_k\}$.

Secondly, for a certain V, the optimal function $\vec{f}$ which minimizes the cost function should satisfy: $\vec{f}(C_k) = \frac{\sum_{(C_k, w_i) \in D_k} VI_i}{|D_k|} = V\vec{m}_k$, where $\vec{m}_k = \frac{\sum_{(C_k, w_i) \in D_k} I_i}{|D_k|}$. That is, $\vec{f}$ should map context C_k to the centroid of the word choice vectors in group D_k. We may therefore rewrite the formula above as:

$$\begin{aligned} & \min_{V} \sum_{1 \leq k \leq N} \sum_{(C_k, w_i) \in D_k} ||V\vec{m}_k - VI_i||^2 \quad \textit{s.t.} |\det(V)| \geq 1 \\ \Leftrightarrow \quad & \min_{V} \sum_{1 \leq k \leq N} \sum_{(C_k, w_i) \in D_k} ||\vec{m}_k - I_i||^2_{V^TV} \quad \textit{s.t.} \det(V^TV) \geq 1 \end{aligned}$$

where the notation $||\vec{t}||_B$ is the Mahalanobis distance: $||\vec{t}||_B = \sqrt{\vec{t}^T B \vec{t}}$. Together, this gives us the exact equation for the minimization of within class distances problem that the RCA algorithm may

solve (Bar-Hillel et al., 2006, p. 945). We therefore directly apply the RCA algorithm to calculate the optimal V: $V = T\hat{R}^{-\frac{1}{2}}$ where T is a constant number and

$$\hat{R} = \sum_{1 \leq k \leq N} \sum_{(C_k, w_i) \in D_k} (I_i - \vec{m}_k)(I_i - \vec{m}_k)^T$$

A.2 APPROXIMATING THE TRANSLATION PROBABILITY FOR REDUNDANT WORD DETECTION

We approximate the translation probability $\log \Pr(f^*|e_-^k)$ with approximations 1 and 2.

We first approximate $\log \Pr(f_*|e_-^k)$ by reusing the alignment structure between e and f_*, as in Approximation 1. To make the alignment structures compatible, we start with redefining e_-^k as $e_1, e_2, \ldots, e_{k-1}, \square, e_{k+1}, \ldots, e_{l_e}$, where the deleted word is left blank. By applying Approximation 1:

$$\begin{aligned}
& \log \Pr(f_*|e_-^k) \\
= \; & \log \sum_a \Pr(a|f_*, e) \frac{\Pr(f_*, a|e_-^k)}{\Pr(a|f_*, e)} \\
= \; & \sum_a \Pr(a|f_*, e) \log \frac{\Pr(f_*, a|e_-^k)}{\Pr(a|f_*, e)} \\
& + \underbrace{\sum_a \Pr(a|f_*, e) \log \left(\Pr(f_*|e_-^k) / \frac{\Pr(f_*, a|e_-^k)}{\Pr(a|f_*, e)} \right)}_{D_{\mathrm{KL}}(a|f_*, e; a|f_*, e_-^k) \approx 0} \\
\approx \; & \sum_a \Pr(a|f_*, e) \log \Pr(f_*|e_-^k, a) + C_2(e)
\end{aligned}$$

We then use IBM Model 1 to calculate $\log \Pr(f_*|e_-^k, a)$, the translation probability under a given alignment structure (Approximation 2).

$$\begin{aligned}
& \sum_a \Pr(a|f_*, e) \log \Pr(f_*|e_-^k, a) \\
= \; & \sum_a \Pr(a|f_*, e) \sum_{1 \leq i \leq l_{f_*}} \log \Pr(f_*^i|e_{-a_i}^k) \\
= \; & \sum_a \Pr(a|f_*, e) \sum_{1 \leq i \leq l_{f_*}} \log \frac{\Pr(f_*^i|e_{-a_i}^k)}{\Pr(f_*^i|e_{a_i})} + C_3(e)
\end{aligned}$$

Note that

$$\log \frac{\Pr(f_*^i|e^k_{-a_i})}{\Pr(f_*^i|e_{a_i})} = \begin{cases} 0 & \text{, for } a_i \neq k \\ \log \frac{\Pr(f_*^i|\square)}{\Pr(f_*^i|e_{a_i})} & \text{, otherwise} \end{cases}$$

$$\begin{aligned} & \sum_a \Pr(a|f_*, e) \sum_{1 \leq i \leq l_{f_*}} \log \frac{\Pr(f_*^i|e^k_{-a_i})}{\Pr(f_*^i|e_{a_i})} \\ = & \sum_a \Pr(a|f_*, e) \sum_{1 \leq i \leq l_{f_*}} I_{a_i=k} \log \frac{\Pr(f_*^i|\square)}{\Pr(f_*^i|e_k)} \\ = & \underbrace{\sum_{1 \leq i \leq l_{f_*}} \underbrace{\Pr(a_i = k|f_*, e)}_{A_{i,k}} \log \frac{\Pr(f_*^i|\square)}{\Pr(f_*^i|e_k)}}_{\text{DIFF}(e^k_-, e)} \end{aligned}$$

Here $A_{i,k} = \Pr(a_i = k \mid f_*, e)$, which is the probability of the i-th word in the translation being aligned to the k-th word in the original sentence.

Through deductions,

$$\begin{aligned} R(k; e) &= \text{LM}(e^k_-) + \text{DIFF}(e^k_-, e) \\ &\quad + C_1(e) + C_2(e) + C_3(e) \end{aligned}$$

the redundancy measure boils down to how we define $\Pr(f_*^i|\square_k)$, which is: when we discard e_k, how do we generate the word it aligns f_*^i with in its translation. This value reflects e_k's contribution in generating f_*^i.

We try to approximate $\Pr(f_*^i|\square_k)$. We note that rare words are often more important, and therefore harder to be generated. We assume $\Pr(f_*^i|\square) = \Pr(e_k|\square) \Pr(f_*^i|e_k)$.

$$\begin{aligned} & \text{DIFF}(e^k_-, e) \\ = & \sum_{1 \leq i \leq l_{f_*}} A_{i,k} \log \frac{\Pr(e_k|\square) \Pr(f_*^i|e_k)}{\Pr(f_*^i|e_k)} \\ = & A(k) \log \Pr(e_k|\square) \end{aligned} \tag{.1}$$

This is the expected number of each word' Chinese alignments (according to $\Pr(a|f_*, e)$), weighted by its importance $\log \Pr(e_k|\square)$. We use e_k's unigram probability to approximate $\log \Pr(e_k|\square)$. This allows us to obtain Equation 4.2.

BIBLIOGRAPHY

Cem Akkaya, Alexander Conrad, Janyce Wiebe, and Rada Mihalcea. Amazon mechanical turk for subjectivity word sense disambiguation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 195–203. Association for Computational Linguistics, 2010.

R.K. Ando. Applying alternating structure optimization to word sense disambiguation. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 77–84. Association for Computational Linguistics, 2006. URL http://acl.ldc.upenn.edu/W/W06/W06-29.pdf#page=93.

Yigal Attali and Jill Burstein. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3), 2006.

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(1):937, 2006.

Ken Beatty. *Teaching and researching: Computer-assisted language learning*. Pearson Education, 2003.

Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

Chris Brockett, William B. Dolan, and Michael Gamon. Correcting ESL errors using phrasal smt techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 249–256, Sydney, Australia, 2006. Association for Computational Linguistics.

Peter F Brown, Vincent J Della Pietra, Robert L Mercer, Stephen A Della Pietra, and Jennifer C Lai. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18 (1):31–40, 1992.

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

Chris Callison-Burch. Fast, cheap, and creative: evaluating translation quality using amazon's mechanical turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural*

Language Processing: Volume 1-Volume 1, pages 286–295. Association for Computational Linguistics, 2009.

Chris Callison-Burch and Mark Dredze. Creating speech and language data with amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 1–12. Association for Computational Linguistics, 2010.

Pi-Chuan Chang, Michel Galley, and Christopher D Manning. Optimizing chinese word segmentation for machine translation performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232. Association for Computational Linguistics, 2008.

Martin Chodorow, Joel R. Tetreault, and Na-Rae Han. Detection of grammatical errors involving prepositions. In *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, SigSem '07, pages 25–30, Prague, Czech Republic, 2007. Association for Computational Linguistics.

James Clarke and Mirella Lapata. Modelling compression with discourse constraints. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1–11, 2007.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34(4):597–614, 2008.

Linguistic Data Consortium et al. English gigaword. *Catalog number LDC2003T05. Available from LDC at http://www. ldc. upenn. edu*, 2003.

Will Coster and David Kauchak. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9, Portland, Oregon, June 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-1601.

D. Dahlmeier and H.T. Ng. Correcting semantic collocation errors with l1-induced paraphrases. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 107–117, Edinburgh, Scotland, UK, July 2011a. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. Grammatical error correction with alternating structure optimization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 915–923, Portland, Oregon, USA, 2011b. Association for Computational Linguistics. ISBN 978-1-932432-87-9.

Daniel Dahlmeier and Hwee Tou Ng. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N12/N12-1067.

Daniel Dahlmeier, Hwee Tou Ng, and Eric Jun Feng Ng. Nus at the hoo 2012 shared task. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pages 216–224, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-2025.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. Building a large annotated corpus of learner english: The NUS corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, 2013.

Robert Dale and Adam Kilgarriff. Helping our own: Text massaging for computational linguistics as a new shared task. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 263–267. Association for Computational Linguistics, 2010.

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

Robert Dale, Ilya Anisimoff, and George Narroway. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pages 54–62, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-2006.

R. De Felice. *Automatic error detection in non-native English*. PhD thesis, University of Oxford, 2008.

John DeNero and Dan Klein. Tailoring word alignments to syntactic machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 17–24, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P07-1003.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 15–24, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-1702.

U. Fischer. *How students learn words from a dictionary and in context*. PhD thesis, Princeton University, 1990.

Jennifer Foster and Oistein E. Andersen. Generrate: Generating errors for use in grammatical error detection. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, EdAppsNLP '09, pages 82–90, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-37-4. URL http://dl.acm.org/citation.cfm?id=1609843.1609855.

Laureen A Fregeau. Preparing ESL students for college writing: Two case studies. *The Internet TESL Journal*, 5(10), 1999.

M. Gamon, J. Gao, C. Brockett, A. Klementiev, W.B. Dolan, D. Belenko, and L. Vanderwende. Using contextual speller techniques and language modeling for ESL error correction. *Urbana*, 51:61801, 2009.

Michael Gamon. High-order sequence modeling for language learner error detection. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 180–189. Association for Computational Linguistics, 2011.

Michael Gamon, Jianfeng Gao, Chris Brockett, Alexandre Klementiev, William B. Dolan, Dmitriy Belenko, and Lucy Vanderwende. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP*, Hyderabad, India, 2008.

Sylvaine Granger, Estelle Dagneaux, and Fanny Meunier. International corpus of learner english. 2002.

Shicun Gui and Huizhong Yang. Zhongguo xuexizhe yingyu yuliaohu.(chinese learner english corpus). *Shanghai: Shanghai Waiyu Jiaoyu Chubanshe*, 2003.

N.R. Han, J. Tetreault, S.H. Lee, and J.Y. Ha. Using an error-annotated learner corpus to develop an ESL/efl error correction system. *LREC, Malta, May*, 2010.

Trude Heift and Mathias Schulze. *Errors and intelligence in computer-assisted language learning: Parsers and pedagogues*. Psychology Press, 2007.

Michael Heilman and Noah A Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019. Association for Computational Linguistics, 2010.

Matthieu Hermet and Alain Désilets. Using first and second language models to correct preposition errors in second language authoring. In *Proceedings of the Fourth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 64–72. Association for Computational Linguistics, 2009.

Hongyan Jing. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315. Association for Computational Linguistics, 2000.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1075096.1075150. URL http://dx.doi.org/10.3115/1075096.1075150.

Kevin Knight and Ishwar Chander. Automated postediting of documents. In *Proceedings of AAAI-94*, Seattle, Washington, 1994.

Kevin Knight and Daniel Marcu. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press, 2000. ISBN 0-262-51112-6. URL http://dl.acm.org/citation.cfm?id=647288.721086.

P. Koehn, F.J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

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

Lillian Lee. Measures of distributional similarity. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 25–32, College Park, Maryland, 1999. Association for Computational Linguistics. ISBN 1-55860-609-3.

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

Mike Levy. Technologies in use for second language learning. *The Modern Language Journal*, 93 (s1):769–782, 2009.

Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-556-8.

Xiaohua Liu, Bo Han, Kuan Li, Stephan Hyeonjun Stiller, and Ming Zhou. SRL-based verb selection for ESL. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1068–1076, Cambridge, Massachusetts, 2010. Association for Computational Linguistics.

Peter Lynch. The origins of computer weather prediction and climate modeling. *Journal of Computational Physics*, 227(7):3431–3444, 2008.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190, Montréal, Canada, June 2012a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N12/N12-1019.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. Exploring grammatical error correction with not-so-crummy machine translation. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pages 44–53, Montréal, Canada, June 2012b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-2005.

Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7. Association for Computational Linguistics, 2002.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, volume 17, pages 591–598, 2000.

Ryan McDonald. Discriminative sentence compression with soft syntactic evidence. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, volume 6, pages 297–304. Association for Computational Linguistics, 2006.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. Mining revision log of language learning SNS for automated japanese error correction of second language learners. In *IJCNLP*, pages 147–155, 2011.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. The conll-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-3601.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-1701.

D. Nicholls. The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 572–581, 2003.

F.J. Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics, 2003.

F.J. Och and H. Ney. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449, 2004.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. 1999.

Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 613–619, Edmonton, Alberta, Canada, 2002. ACM. ISBN 1-58113-567-X.

Y.A. Park and R. Levy. Automated whole sentence grammar correction using a noisy channel model. *Proceedings of ACL 2011*, June 2011.

Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, ACL '93, pages 183–190, Columbus, Ohio, 1993. Association for Computational Linguistics.

A. Rozovskaya and D. Roth. Training paradigms for correcting errors in grammar and usage. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 154–162. Association for Computational Linguistics, 2010a.

Alla Rozovskaya and Dan Roth. Annotating ESL errors: Challenges and rewards. In *Proceedings of the NAACL HLT 2010 fifth workshop on innovative use of NLP for building educational applications*, pages 28–36. Association for Computational Linguistics, 2010b.

Alla Rozovskaya and Dan Roth. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 961–970, Cambridge, Massachusetts, 2010c. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. Training paradigms for correcting errors in grammar and usage. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 154–162, Los Angeles, California, 2010d. Association for Computational Linguistics. ISBN 1-932432-65-5.

Alla Rozovskaya, Mark Sammons, Joshua Gioja, and Dan Roth. University of illinois system in hoo text correction shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 263–266. Association for Computational Linguistics, 2011.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. The university of illinois system in the conll-2013 shared task. In *In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared*. Citeseer, 2013.

Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 614–622. ACM, 2008.

Matthew G Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. TER-Plus: paraphrase, semantic, and alignment enhancements to translation edit rate. *Machine Translation*, 23(2-3): 117–127, 2009.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 254–263. Association for Computational Linguistics, 2008.

Andreas Stolcke. Srilm-an extensible language modeling toolkit. In *Proceedings of the international conference on spoken language processing*, volume 2, pages 901–904, 2002.

Ben Swanson and Elif Yamangil. Correction detection and error type selection as an ESL educational aid. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 357–361, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N12-1037.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 198–202. Association for Computational Linguistics, 2012.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. Using parse features for preposition selection and error detection. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 353–358, Uppsala, Sweden, 2010a. Association for Computational Linguistics.

Joel R. Tetreault and Martin Chodorow. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 865–872, Manchester, United Kingdom, 2008. Association for Computational Linguistics. ISBN 978-1-905593-44-6.

Joel R Tetreault, Elena Filatova, and Martin Chodorow. Rethinking grammatical error annotation and evaluation with the amazon mechanical turk. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 45–48. Association for Computational Linguistics, 2010b.

Jason Gordon Williams. Providing feedback on ESL students' written assignments. *The Internet TESL Journal*, 4(10), 2003.

Yuanbin Wu and Hwee Tou Ng. Grammatical error correction using integer linear programming. *Proceedings of ACL 2013*, August 2013.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics, 2012.

Huichao Xue and Rebecca Hwa. Syntax-driven machine translation as a model of ESL revision. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1373–1381. Association for Computational Linguistics, 2010.

Huichao Xue and Rebecca Hwa. Improved correction detection in revised ESL sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, MD, USA, 2014. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics, 2011.