A Statistical Model of Error Correction for Computer Assisted Language Learning Systems

Halizah Basiron

a thesis submitted for the degree of
Doctor of Philosophy
at the University of Otago, Dunedin, New Zealand.

May 24, 2012
Abstract

This thesis presents a study in the area of computer assisted language learning systems. The study focusses on the topic of automatic correction of student errors. In the thesis, I will describe a novel statistical model of error correction, trained on a newly gathered corpus of language data from Malaysian EFL learners, and tested on a different group of students from the same population.

The main novelty of my statistical model is that it explicitly represents ‘corrections’—i.e. circumstances where a language teacher corrects a student’s language. Most statistical models used in language learning applications are simply models of the target language being taught; their aim is just to define the kinds of sentence which are expected in the target language. These models are good at recognising when a student’s utterance contains an error: any sentence which is sufficiently improbable according to a model of the target language can be hypothesised to contain an error. But they are not so good at providing suggestions about how to correct errors. In any student’s sentence, there are many things which could be changed: the space of possible corrections is too large to be exhaustively searched. A statistical system which explicitly models the incidence of corrections can help guide the search for good corrections. The system which I describe in this thesis learns about the kinds of context in which particular corrections are made, and after training, is able to make quite good suggestions about how to correct sentences containing errors.
Acknowledgements

This thesis would not have been possible without the guidance and the help of several individuals who contributed and extended their valuable assistance in the preparation and completion of this study.

First and foremost, my utmost gratitude to my two supervisors, Associate Professor Alistair Knott and Associate Professor Anthony Robins. Thank you for your knowledge, advice, guidance, and patience. You both have been my inspiration as I hurdle all the obstacles in the completion this research study.

Thank you from the bottom of my heart to my beloved husband Nurulhalim Hassim, and my three adorable children Aini, Ahmad, and Iffah for your selfless love and support through this all.

Thank you to my mother-in-law Noraini Shafei, my father Basiron Abd. Rahman, my stepmother Dzabedah, my sisters, my brothers, my in-laws, my nieces, my nephews and all my relatives for your love and encouragement.

Thank you to all those in Computer Science Department, especially the technical staff, for your help while I am in the department.

Thank you to UTeM and MOHE which have sponsored my study here.

To all my friends, Lyn, Nija, Nida, Dila, Maryam, Lisa, Nur Ain, Afeiz, Wira, and Suzanne (to name a few), thank you for lending me your shoulder to cry on.

Finally, thank you to others who have helped me in the completion of this study.
3 An empirical study of learners’ errors in written language learning dialogues

3.1 Introduction .................................................. 65
3.2 Format of the Study .......................................... 67
3.2.1 Subjects ..................................................... 67
3.2.2 Materials and Methods .................................. 69
3.3 Existing Error Classification Schemes ..................... 71
3.3.1 The Cambridge Learner Corpus ......................... 72
3.3.2 The National Institute of Information and Communications Technology Japanese Learner of English .................. 73
3.3.3 The FreeText System ..................................... 74
3.3.4 Spelling Correction Techniques ....................... 75
3.4 My Error Classification Scheme ......................... 76
3.4.1 Effectiveness of Error Classification Schemes .......... 79
3.4.2 Error Tag Categories ..................................... 80
3.4.2.1 Agreement Errors ..................................... 80
3.4.2.2 Tense Errors .......................................... 81
3.4.2.3 Spelling Errors ........................................ 81
3.4.2.4 Vocabulary Errors .................................... 81
3.4.2.5 Delete/Insert/Substitute/Transpose Tags ............. 83
3.4.2.6 Dialogue Errors ........................................ 86
3.4.2.7 Unclassifiable Error Tags ............................ 87
3.4.2.8 Grammatical Responses .............................. 87
3.5 Errors Correction and Annotation ......................... 88
3.5.1 Provision of Correct Models ......................... 88
3.5.1.1 Ambiguous Utterances ............................. 88
3.5.2 Annotating Tasks and Order of correction ............ 88
3.6 Inter-coder Reliability Tests ............................... 89
3.6.1 Percent Agreement Test ............................... 91
3.6.2 The Kappa Reliability Test ........................... 92
3.6.3 The Alpha Reliability test ............................ 94
3.7 Assessing Inter-Coder Agreement ......................... 96
3.7.1 Coders for the Reliability Tests ....................... 97
3.7.2 Prior to the Reliability Tests Process ........................................ 97
3.7.3 Measuring Inter-coder Reliability ........................................... 98
3.7.4 Results of the reliability tests ................................................ 103
3.8 Results of the Corpus Data Analysis .......................................... 105
  3.8.1 Most Common Errors .......................................................... 105
    3.8.1.1 Omission of determiner and the copula be ....................... 107
    3.8.1.2 Incorrect use of Tense ............................................... 109
    3.8.1.3 Discussion .............................................................. 111
  3.8.2 Longitudinal Comparison .................................................... 111
    3.8.2.1 General Differences between Form 1 and Form 3 of Sch1 school students ......................................................... 112
    3.8.2.2 Difference in the distribution of errors between Form 1 and Form 3 ................................................................. 114
    3.8.2.3 Discussion .............................................................. 120
3.9 Summary ..................................................................................... 122
4 A Statistical Model of Error Correction ........................................ 125
  4.1 Error Correction Module in The Kaitito System ............................ 126
    4.1.1 Interpretation and Disambiguation Module ......................... 127
    4.1.2 Perturbation Module ........................................................ 128
  4.2 Language Modelling ................................................................. 130
    4.2.1 Maximum Likelihood Estimate .......................................... 132
    4.2.2 Smoothing ........................................................................ 134
      4.2.2.1 Add-One Smoothing .................................................. 135
      4.2.2.2 Witten-Bell Smoothing .............................................. 136
    4.2.3 Backoff ............................................................................ 140
  4.3 The Proposed Model of Error Correction .................................... 145
    4.3.1 The Model ........................................................................ 147
    4.3.2 An Example of The Model ................................................ 151
  4.4 Development of A $N$-gram Perturbation Corpus ......................... 154
    4.4.1 Generating trigram perturbations ...................................... 157
      4.4.1.1 Word-insertion function .......................................... 160
      4.4.1.2 Word-deletion function .......................................... 161
      4.4.1.3 Word-substitution function .................................... 162
      4.4.1.4 Word-transposition function ................................... 164
    4.4.2 Adjacent Errors ................................................................ 164
    4.4.3 Generating bigram and unigram perturbations ...................... 165
    4.4.4 Counting $n$-gram perturbations ....................................... 167
  4.5 Generating feasible perturbations ............................................. 168
    4.5.1 Insertion .......................................................................... 168
    4.5.2 Deletion .......................................................................... 169
    4.5.3 Substitution ...................................................................... 169
    4.5.4 Transposition ................................................................. 170
  4.6 An Evaluation of the Proposed Model ....................................... 170
    4.6.1 The Model to be Evaluated .............................................. 170
    4.6.2 Data ................................................................................. 171
4.6.3 Parameters
4.6.3.1 Error model 1: Comprehensive
4.6.3.2 Error model 2: PN-label-backoff
4.6.3.3 Building separate error models for different sentence types
4.6.4 Evaluation Procedure
4.6.4.1 Data Partitioning
4.6.4.2 Data Analysis
4.6.4.3 Data Validation

4.7 Results and Discussion
4.7.1 Result for Comprehensive error model
4.7.2 Result for PN-label-backoff error model
4.7.2.1 Building a separate error model from PN-label-backoff error model
4.7.3 Result for Tense error model
4.7.3.1 Present tense
4.7.3.2 Past-tense and Future-tense
4.7.4 Result for Pronoun error model
4.7.5 Result for Question Types error model
4.7.6 Discussion

4.8 Summary

5 Implementation of a Practical Dialogue-based CALL System
5.1 Building a Grammar for the Domain
5.1.1 Parsing with the ERG
5.1.2 Generating a reduced version of ERG
5.2 A Simple Dialogue Manager for Kaitito
5.2.1 The Basic Structure of A Dialogue Session
5.3 The Teaching Response Module
5.3.1 The Perturbation Module
5.3.2 Parsing Candidate Perturbations
5.3.3 The Form of Correction Suggestions: Teaching Hints
5.3.4 A Default Response
5.4 A Sample Session with Kaitito
5.5 Programming Issues
5.6 Summary

6 Evaluation of the Error Correction Model
6.1 Methods
6.1.1 Participants
6.1.2 Procedures
6.1.3 Materials and System Usage
6.1.4 Data Captured during the Evaluation
6.2 Results
6.2.1 Summary of Students’ Responses
6.2.2 Preliminary Checking
6.2.3 Evaluating Grammar ................................................. 232
6.2.4 Evaluating The Error Correction System ....................... 234
  6.2.4.1 Evaluating the perturbation algorithm ....................... 235
  6.2.4.2 Evaluating the provision of teaching hints .................. 237
6.3 Discussion ............................................................... 240
6.4 Summary ................................................................. 241

7 Conclusion ...................................................................... 243
  7.1 Contributions ............................................................. 243
  7.2 Future Work ............................................................... 245
  7.3 Final Remark ............................................................. 246

References ......................................................................... 249

A Other Classifications for Corrective Feedback ...................... 265
B Teaching and Learning English in Malaysia ......................... 267
C The Questions featuring in the Student Questionnaire .......... 273
  C.1 Present Tense Questions ............................................. 273
  C.2 Past Tense Questions ................................................. 276
  C.3 Future Tense Questions .............................................. 278
D Tags of Error Categories .................................................. 281
E A Complete Set of Sentences used to develop the reduced-ERG 283
F A Listing of the Additional Malaysia-specific Proper Names added to Kaitito’s lexicon ..................................... 287
G Students’ Transcript Samples ........................................... 291
  G.1 Sample 1 .................................................................. 291
  G.2 Sample 2 .................................................................. 293
  G.3 Sample 3 .................................................................. 294
List of Tables

2.1 Error analysis by PDA .............................................. 17
2.2 Explicit and Implicit corrective feedback ............................ 34
2.3 The tractability of different kinds of CF. .......................... 40
2.4 Word categories for English. ...................................... 43
2.5 Phrase labels for English. ......................................... 43
2.6 Context free grammar rules ....................................... 45
2.7 (Error) Grammar rules .............................................. 46
2.8 A NP rule embedded with constraint relaxation rules .......... 48
2.9 Probabilistic context free grammar rules with a probability value (in bracket) assigned to each rule .......................... 50
2.10 Types (dialogue-based or parser-based) of CALL systems .... 63
2.11 Types of corrective feedback provided by CALL systems .... 64
3.1 Numbers of students, by school, form and stream ............... 68
3.2 A composition and percentage across races ....................... 68
3.3 The Types of Pronoun ............................................. 69
3.4 The Sample Data .................................................. 70
3.5 Common error types identified by Kukich (1992) ................ 76
3.6 The Agreement Error Tags ....................................... 81
3.7 Error annotation using \textit{sva, det-n-ag,} and \textit{noun-num-err} tags .............. 82
3.8 Error annotation using \textit{tense-err}(X,Y) tags .................. 82
3.9 Wrongly spelled word types ....................................... 83
3.10 The list of linguistic forms and its respective symbol used .. 84
3.11 Error annotation using delete/insert/substitute/transpose tags 85
3.12 Error annotation using dialogue error tags .................... 86
3.13 Sentences annotated with the unclassifiable error tags ....... 87
3.14 Grammatical Sentences .......................................... 87
3.15 Ambiguous responses ............................................ 88
3.16 Annotation process for ungrammatical responses ............... 90
3.17 A example of inter-coder agreement table ...................... 91
3.18 The $\kappa$ values and strength of agreement level ............. 93
3.19 The range of $\alpha$ values and its interpretation .............. 96
3.20 Order of error tags. ............................................. 98
3.21 Sequence of error tags. ......................................... 99
3.22 Error tags realignment. ......................................... 99
3.23 Limited number of error tags. .................................. 100
3.24 Equivalent predicates in a same sequence order. .............. 100
4.19 Total number of trigram perturbations generated from each error model 193

5.1 Groups of sentences in the learner corpus .............................. 198
5.2 Parsing results for Orig-crt-same-PN and One-err-sentce using the ERG .................................................. 198
5.3 Comparison of parsing results between Orig-crt-same-PN and One-err-sentce using full ERG and reduced-ERG ............................ 202
5.4 Results for accuracy, precision and recall of the reduced-ERG grammar 203
5.5 The Total Numbers of lexical added in the reduced-ERG ............ 203
5.6 The Percentage of Source Code Lines I Contributed .................. 216

6.1 Dialogue ids created per class ............................................ 227
6.2 Total of student answers provided ..................................... 227
6.3 A sample of preliminary checking ........................................ 230
6.4 Distribution of percentage on each preliminary checking case ...... 231
6.5 Total of responses after preliminary checking done .................. 231
6.6 Results for the analysis of students’ answers .......................... 232
6.7 Results for accuracy, precision and recall of the reduced-ERG grammar 233
6.8 Results of the performance for ErrorError sentences .............. 236
6.9 Results of the performance for one-err sentences .................. 236
6.10 Results of the performance of error correction model evaluated on a training corpus .................................................. 236
6.11 Results of the performance for useful teaching hints ............... 240
6.12 The Overall Results of the Evaluation ................................. 241

A.1 Categories of corrective feedback as defined by Ferreira .......... 265
A.2 Categories of corrective feedback as claimed by Ellis ............. 266
B.1 Summary of Malaysian school levels and ages of students ....... 268
D.1 Error Categories ......................................................... 282
List of Figures

2.1 The architecture of Kaitito ........................................... 21
2.2 Psycholinguistic sources of errors (adapted from Ellis (1994, pg 58)) .. 29
2.3 A parse tree for “The boy chased the girl.” ............................. 43
2.4 Two parse trees for a sentence “The man fed her fish food” ................. 44
2.5 A parse tree for mal-rules in (a) and correct rules in (b). ...................... 47
2.6 Two parse trees for a sentence “He fed her fish food” .......................... 51

3.1 Distribution of percentage of each error category .......................... 106
3.2 Percentage of a word missing in the ins($X$) error tag category ............. 106
3.3 Percentage of a word missing in the ins($X$) error tag category for the Sch1 school .......................................................... 108
3.4 Percentage of tense error category distribution ............................... 109
3.5 Percentage of tense error category distribution for the Sch1 school only 110
3.6 Average of total errors committed per student (with standard error bars) 113
3.7 Average of total words produced per student (with standard error bars) 113
3.8 Average of total each error category done per student for each group 114
3.9 A flowchart to show which significant tests are performed .......................... 116

4.1 A sentence interpretation process in Kaitito(from van der Ham (2005)) 127
4.2 Probability of seen, low count and unseen events before and after smooth- ing technique ......................................................... 134
4.3 Process for Development of a N-gram Perturbation Corpus .................. 155
4.4 Location of unmatched and matched words at cur_loc in the original sentence ................................................................. 160
4.5 A sample of a perturbation list .................................................. 177
4.6 Average performance for the Comprehensive error model for each training data set ................................................................. 179
4.7 Comparison results between Type 1 and Type 2 on a same test input sentence. ................................................................. 180
4.8 Comparison of average performance between Comprehensive and PN- label-backoff error models .............................................. 181
4.9 Comparison of average performance among three error models ............. 183
4.10 Comparison of average performance between Present-tense and Comprehen- sive error models ................................................................. 185
4.11 Comparison of average performance between Tense and Comprehensive error models ................................................................. 187
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>Computer Assisted Language Learning</td>
<td>2</td>
</tr>
<tr>
<td>CF</td>
<td>Corrective Feedback</td>
<td>3</td>
</tr>
<tr>
<td>CFG</td>
<td>Context Free Grammar</td>
<td>44</td>
</tr>
<tr>
<td>CLC</td>
<td>Cambridge Learner Corpus</td>
<td>72</td>
</tr>
<tr>
<td>CMC</td>
<td>Computer-Mediated Communication</td>
<td>10</td>
</tr>
<tr>
<td>DOP</td>
<td>Data-Oriented Parsing</td>
<td>50</td>
</tr>
<tr>
<td>EFL</td>
<td>English as a Foreign Language</td>
<td>1</td>
</tr>
<tr>
<td>ERG</td>
<td>English Resource Grammar</td>
<td>20</td>
</tr>
<tr>
<td>ESL</td>
<td>English as a Second Language</td>
<td>52</td>
</tr>
<tr>
<td>FRIDA</td>
<td>French Interlanguage Database</td>
<td>74</td>
</tr>
<tr>
<td>HPSG</td>
<td>Head Phrase Structure Grammar</td>
<td>20</td>
</tr>
<tr>
<td>ICICLE</td>
<td>Interactive Computer Identification and Correction of Language Errors</td>
<td>24</td>
</tr>
<tr>
<td>L1</td>
<td>First or Native Language</td>
<td>28</td>
</tr>
<tr>
<td>L2</td>
<td>Second or Target Language</td>
<td>29</td>
</tr>
<tr>
<td>LKB</td>
<td>Linguistic Knowledge Building</td>
<td>20</td>
</tr>
<tr>
<td>LM</td>
<td>Language Modelling</td>
<td>55</td>
</tr>
<tr>
<td>MASI</td>
<td>Measuring Agreement on Set-valued Items</td>
<td>94</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum Likelihood Estimate</td>
<td>132</td>
</tr>
<tr>
<td>NICT JLE</td>
<td>The National Institute of Information and Communications Technology Japanese Learner English</td>
<td>73</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
<td>5</td>
</tr>
<tr>
<td>PCFG</td>
<td>Probabilistic Context Free Grammar</td>
<td>50</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>POS</td>
<td>Part-Of-Speech, page 50</td>
<td></td>
</tr>
<tr>
<td>SLA</td>
<td>Second Language Acquisition, page 10</td>
<td></td>
</tr>
<tr>
<td>SMT</td>
<td>Statistical Machine Translation, page 56</td>
<td></td>
</tr>
<tr>
<td>SPELL</td>
<td>Spoken Electronic Language Learning, page 17</td>
<td></td>
</tr>
<tr>
<td>SST</td>
<td>Standard Speaking Test, page 73</td>
<td></td>
</tr>
<tr>
<td>SVA</td>
<td>Subject-Verb Agreement, page 120</td>
<td></td>
</tr>
<tr>
<td>WB</td>
<td>Witten-Bell, page 136</td>
<td></td>
</tr>
<tr>
<td>WWW</td>
<td>World Wide Web, page 55</td>
<td></td>
</tr>
<tr>
<td>XML</td>
<td>eXtension Markup Language, page 72</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This thesis presents a study in the area of computer assisted language learning systems. The study focuses on the topic of automatic correction of student errors. In the thesis, I will describe a novel statistical model of error correction, trained on a newly gathered corpus of language data from Malaysian English as a foreign language (EFL) learners, and tested on a different group of students from the same population. My research focuses on a statistical model of error correction in computer assisted language learning systems.

The main novelty of my statistical model is that it explicitly represents ‘corrections’—i.e. circumstances where a language teacher corrects a student’s language. Most statistical models used in language learning applications are simply models of the target language being taught; their aim is just to define the kinds of sentence which are expected in the target language. These models are good at recognising when a student’s utterance contains an error: any sentence which is sufficiently improbable according to a model of the target language can be hypothesised to contain an error. But they are not so good at providing suggestions about how to correct errors. In any student’s sentence, there are many things which could be changed: the space of possible corrections is too large to be exhaustively searched. A statistical system which explicitly models the incidence of corrections can help guide the search for good corrections. The system which I describe in this thesis learns about the kinds of context in which particular corrections are made, and after training, is able to make quite good suggestions about
how to correct sentences containing errors.

To begin the thesis, first I will talk about the motivations of the research that led me to the topic of automated error correction (§1.1). Then I will summarise the research questions to be addressed in the thesis in §1.2. I end this introductory chapter with the structure of my thesis as described in §1.3.

1.1 Motivation

A computer assisted language learning system (CALL) is a computer application used as a supplementary material in language learning and teaching. CALL systems have focussed on several different language skills: for instance, grammar, vocabulary learning, pronunciation, reading, writing, listening, and speaking. Some systems target a specific language skill and some target multiple skills (Stockwell, 2007).

In this thesis, I will focus on systems which interact with student users in the form of a dialogue. These systems allow students to practise language in a naturalistic context: normally, when we use language, we do so in the context of a dialogue. Online chat systems are a simple example of how technology can support dialogue-based language teaching (Loewen and Erlam, 2006). But these systems assume the presence of a human teacher. In the systems I will focus on, the role of the teacher in the dialogue is played by a computer. These systems are called dialogue-based CALL systems. Using such a system, learners can practise their speaking skill without the presence of a human teacher.

The learners communicate with a computer which acts as their teacher. Similarly to a chat system, the learners can use the dialogue-based CALL at any time they wish. Although there are some dialogue-based systems that require an Internet connection, such a system can in principle run on a stand alone basis. A dialogue-based CALL system is particularly suitable for learners who have low confidence in communicating the learnt language in a normal classroom environment.

In a normal language learning classroom, when students make mistakes, the teacher typically responds with an intervention to correct the mistakes. Such responses are known as corrective feedback. The feedback can be a correct sentence, an explanation
of grammatical rules, and/or hints for an ill-formed sentence, which is delivered in a verbal, written or signal form. Lyster and Ranta (1997) categorised 6 different types of corrective feedback: explicit correction, recast, clarification requests, metalinguistic, elicitation and repetition. The main objective of the provision of corrective feedback (CF) is to make the learners aware of mistakes or errors they have committed. Moreover, research studies have shown that the provision of CF helps language learners in acquiring the target language. See Basiron, Knott, and Robins (2008), Kim (2004), Suzuki (2004), and Tatawy (2002) for some example studies.

Research studies on the effectiveness of CF provision have been carried out in different environments of language learning: normal classrooms (Lyster and Ranta, 1997; Ellis, Loewen, and Erlam, 2006), online chatting systems (Loewen and Erlam, 2006), and CALL systems (Nagata, 1997; Heift, 2004; Ferreira, 2006).

First I will consider language learning in the classroom. Lyster and Ranta (1997) report recast feedback is the most frequently used by teachers but it is the least effective in assisting learners during language learning and teaching. Ellis, Loewen, and Erlam (2006) conclude learners who receive metalinguistic feedback, outperform learners who receive recast feedback. A recast feedback utterance is an utterance which is a correct version of an ill-formed utterance. A metalinguistic feedback utterance is an utterance which consists of a grammatical description of an ill-formed utterance.

Second is learning a language using an online chatting system. Loewen and Erlam (2006) replicate the experiment done by Ellis et al. (2006) to compare the effectiveness between recast and metalinguistic feedback provided to two separate groups while using the online chatting system. However, in terms of the effectiveness of recast and metalinguistic feedback, results yielded no significant difference in performance between the recast and metalinguistic groups.

Third is language learning using CALL systems. Heift (2004); Nagata (1997); Ferreira (2006) carried out an experiment to investigate the effectiveness of various types of CF provided to learners while using the CALL systems. Overall, results show the learners who receive metalinguistic feedback perform better than the learners who receive other CF types. Although the metalinguistic feedback proved to be beneficial in language learning, this feedback is not frequently used by language teachers. More-
over, Ellis (2005) claims a provision of metalinguistic feedback discourages learners from constructing new sentences in language learning. Lyster and Ranta (1997) claim language teachers prefer to provide recast than metalinguistic especially for beginner learners. MacKey and Philp (1998) also argue that the provision of recasts have proved to be beneficial to beginners.

Initially, I was interested to investigate the effectiveness of metalinguistic and recast feedback provided in a CALL system. The CALL is a dialogue-based CALL which is targeted to English learners to practise their conversation skill on selected given topics. In order to provide such feedback, I gathered a language corpus from a group of Malaysian high-school EFL students. (I am Malaysian, hence my interest in this particular group of students.) The corpus consists of students’ responses to some typical questions in daily conversations. I carried out an error analysis on the corpus to gain an idea of what are the most common errors committed by the students. My original idea was to identify the types of error which occurred most frequently, and develop an error correction system specifically targeting these common error types. However, when I analysed the corpus, there were no obvious ‘most frequent’ categories of error. Since the corpus consists of many types of error, it is hard for me to only focus on a specific type of word error. This reason has led me to search for an alternative to the original plan.

The corpus of learner data that I gathered is a very rich source of information. It is not only the learners’ personal data, but also has other valuable contents, such as the learners’ responses, both grammatical and ungrammatical, and the proposed corrections for each ungrammatical sentence. Using a database of erroneous sentences and their proposed corrections, I realised it was possible to develop a “surface-based” model of error correction. The model would statistically describe how an erroneous sentence can be corrected based on common surface transformations of erroneous sentences. The transformations are *word deletion*, *word insertion*, *word substitution* and *word transposition*.

The use of Natural Language Processing (NLP) techniques in CALL has proven useful in language learning (Heift and Schulze, 2007). Nerbonne (2002) claims most of the work in CALL which implements NLP technologies is concerned with correction of
grammatical errors. Error correction for an erroneous sentence is a way to produce a reconstruction of the sentence to become an error-free sentence. There are two basic techniques of error correction (Sun, Liu, Cong, Zhou, Xiong, Lee, and Lin, 2007): *symbolic grammar-based* and *statistical-based* methods. The former defines a set of grammar rules and if a grammatical structure of the sentence isn’t available in the grammar rules, the sentence is considered ungrammatical. The statistical techniques provide probability scores to rank a level of acceptance for the grammatical structure of the sentence.

When designing an error correction system, it is important that it provides educationally useful responses to students. This helps the learners to understand the errors they made, and work on improving their erroneous sentences. Hence, it is essential to develop a CALL system which targets certain types of error or covers a particular context for the language learners. In order to support the learners in learning from the errors they made, error correction utterances must be easily understood by the learners. There is a lack of research focus on this issue. Lee (2009) also highlights that the central problem of providing a correction for an erroneous sentence in his error correction system is to “determine the most appropriate word, given its context within the sentence.” An appropriate correction for a particular erroneous sentence can be provided if a list of pairs of learners’ ill-formed sentences and their suggested corrections is available. The student corpus which I gathered has exactly the right form: it consists of pairs of ill-formed sentences and their corrections. Of course, to use this corpus, I must develop a system which can learn in which linguistic contexts particular corrections are offered. Therefore, a model of an error correction method, that is able to provide suggested corrections which are appropriate and easily understood by language learners is needed.

### 1.2 Research Objectives

For the reasons just mentioned, I would like to propose a statistical model of error correction for CALL systems. The research I present in this thesis has two objectives. The first is to develop a model of error correction. The second objective is to investi-
gate how well the correction model provides appropriate corrections for ungrammatical sentences in real students.

1.3 Overview of The Thesis

This thesis consists of seven chapters including this introductory chapter. Firstly, in Chapter 2, I start with some background studies of current issues in CALL especially that are related to my research.

Second, an empirical study is conducted. The foundation for this study is the empirical study of learners’ errors which I mentioned above. This study is described in detail in Chapter 3. The outcome of the study is a learner corpus.

In Chapter 4, I will describe the statistical model of error correction that I propose. The model is based on statistical language modelling (LM) techniques: *Witten-Bell* smoothing and *Katz Backoff*. The learner corpus created during the empirical study is used to evaluate the error correction model. I used n-fold cross-validation to evaluate the performance of the model using the learner corpus as the training data.

Then, in Chapter 5, I will explain the implementation of a practical error correction system. The system is designed to be used during a language learning dialogue, and to incorporate the statistical model of error correction that I proposed. The model is implemented within the *Kaitito* system. *Kaitito* is the dialogue-based CALL system developed by Dr. Alistair Knott at the University of Otago.

I will describe the evaluation of the practical dialogue-based CALL system in Chapter 6. Participants in the evaluation were Malaysian learners of English as a second or foreign language. The main objective of the evaluation was to examine the performance of my statistical error correction model in providing appropriate corrections for ill-formed sentences. I will also discuss results of the evaluation from an analysis I performed on transcripts of conversations between the system and the learners.

Finally my thesis concludes in Chapter 7 with a brief overview of my research contributions, and an identification of potential areas for future work.
Chapter 2

Literature Review

My thesis focusses on automatic error correction techniques in a dialogue-based computer assisted language learning (CALL) system. This chapter provides a review of literature about this topic. Firstly, I start with an introduction to CALL and its development phases in §2.1. Then in §2.2, I relate the phases to theories of second language learning. Later, I discuss two types of CALL systems that are cross referenced to each other. The systems are dialogue-based CALL systems (§2.3) and parser-based CALL systems (§2.4). After that, I describe grammatical errors committed by language learners during language learning (§2.5). Following this, in §2.6 I describe various types of response that language teachers provide to explain how their learners’ errors can be corrected. The responses are called corrective feedback (CF). In particular I focus on studies that investigate the effectiveness of CF provided in various of language learning setting (§2.6.3). Eventually in §2.7, I give a survey of automatic error correction techniques focused on two approaches: grammar-based (§2.7.1) and statistical methods (§2.7.2). Some issues about these techniques are addressed in §2.7.3. Finally, some conclusions and my research directions are highlighted in §2.8.

2.1 Computer Assisted Language Learning

Computer assisted language learning is an area in which a computer is used as supplementary material in language learning and teaching. Here, my interest is more to
computer-assisted second or foreign language learning. A traditional way of language learning and teaching is where a teacher is standing in front of her/his students giving language materials in a classroom environment. The materials can be presented verbally, or in a written form, for instance on a whiteboard. Computers provide a new technology, which assists teachers in their language teaching.

According to Warschauer (1996), there have been three development phases of CALL that characterise how computers are used during language learning: behaviouristic, communicative, and integrative. Warschauer claims a commencement of a new phase is neither related to proposing new technologies nor opposing technologies used in the previous phase. Heift and Schulze (2007) claim the first two phases of CALL defined by Warschauer (1996) are influenced by which language teaching and learning theories were prevalent at that time.

The first phase, behaviouristic CALL systems, were introduced based on the Behavioural theory of learning (Skinner, 1985). The theory was dominant in language teaching and learning between 1960s and 1970s. The basic idea of the theory is humans learn when they are asked to perform a small task repetitively. This is called a drill and practice concept.\(^1\) The concept means a small topic of a certain language skill i.e. a grammar of definite and indefinite determiners, is taught to language learners, then the learners repetitively do exercises of the grammar until the learners gain a good hand at the grammar.

In the behaviouristic phase, CALL systems were developed to act as a simple machine which delivers learning materials to language learners. The drill and learning concept is parallel to the nature of computers, in which computers are machines that can perform tasks repetitively. Therefore, language learning is more effective when the same learning materials are provided and practised by the learners repetitively for a period of time (Warschauer, 1996). Also, while teachers have constraints on availability and time, computers are available to access for unlimited time.

An example of a drill and practice CALL system is PLATO (Programmed Logic for Automatic Teaching Operations) system ((Ahmad, Corbett, Rogers, and Sussex, 1985)

---

\(^1\)Further information about the behavioural theory of learning is described in §2.2.1.
as cited in (Warschauer, 1996)). Teachers use PLATO to design instructional materials such as how to learn the grammar involving singular and plural nouns. Then language learners use PLATO to learn, do exercises and develop skill on the learnt grammar. However, the behaviouristic CALL systems became less popular when the Behavioural theory lost favour by language teachers. Exposure to other computer applications that can be used in language teaching and learning, added to this decline.

Secondly, communicative CALL systems were implemented when a new language teaching and learning approach called the Communicative approach was introduced and became popular between 1970s and 1980s. The idea of this approach is to encourage communication between teachers and learners by using real-life examples as teaching materials.²

In the communicative phase, the CALL systems act as a tool or a stimulus. Firstly, using CALL systems as a tool means to utilise other software such as word processors, or spelling and grammar checkers to assist learners to gain more understanding about a target language. Secondly, using a CALL system as a stimulus means language learners use a computer to help them to collect information and use the information to solve problems given by language teachers. In the communicative approach, teaching and learning language grammar rules are not the main focus but rather how a language is used in a specific situation. Therefore, beside an implementation of CALL systems that are specifically developed to facilitate mastery of a certain language skill, other computer applications which are purposely developed to solve a certain problem are also introduced. An example is The SimCity application. The SimCity is not a computer application developed specifically to learn a language but rather a computer simulator game application with the aim of setting up a city. In this game, the purpose is to design a city. Learners need to build houses, a shopping complex, or gardens, and also set up transportation systems. Therefore, whilst playing the games, teachers ensure that the learners have to use sentences and vocabularies that are suitable to the purpose of the game.

The third phase is integrative CALL systems that were introduced in 1990s. Here,

²The approach is related to the Output and Interaction Hypothesis as described in §2.2.3 and §2.2.4 respectively.
two new technologies are implemented in CALL development: multimedia and the Internet. Multimedia elements consist of a variety of media such as text, graphics, sound, animation, and video. By incorporating these elements in CALL, the learner can read, write, speak and listen in a single activity, just like in the real world. Thus, it creates more authentic learning environments, integrates skills easily, and helps the learners to obtain greater control over their learning. Multimedia technology, as well as the emergence of Internet technology, obviously have created a new learning environment. Learners can communicate with other learners or their teacher 24 hours a day from school, work, or home. The use of multimedia and the Internet in a CALL system is known as Computer-Mediated Communication (CMC) system. Examples of CMC systems are online chat such as Internet Relay Chat, Yahoo! Messenger, Skype and Google Talk, to name a few. While online chat systems require an interaction between humans, dialogue-based CALL systems enable an interaction between language learners and a computer. I will refer back to dialogue-based CALL systems in §2.3.

The introduction of integrative CALL systems is not driven by a new learning theory, but rather by the emergence of multimedia and internet technologies. In this phase also, the two theories of language learning can be practised. In the next section, I will describe some of the theoretical approaches to language learning, which have influenced CALL research.

2.2 Review on Learning and Second Language Learning Theories

Currently, CALL systems are sophisticated multimedia products, which support a wide range of language teaching exercises. In 1997, Chapelle suggested that the designers of CALL systems should look more toward theories of second language acquisition (SLA). Since then the developers of CALL systems have taken into account pedagogical principles (Chapelle, 2005). I will use the terms “second language acquisition” and “second language learning” interchangeably. Recently some CALL designers have taken SLA theory into account for implementing a pedagogical approach in developing CALL
software. There are many theories of second language acquisition formulated. The following sections describe several theories of language teaching and learning. However, the theories that are most relevant to my research are those which relate to the role of interaction/dialogue in language learning. This is because I would like to relate these theories to corrective feedback research that I will discuss in §2.6.

2.2.1 The Behavioural Theory of Learning

The theory was first proposed by Skinner (1985). In this theory, a learning process takes place automatically when a human performs a task repetitively. This concept is known as drill and practice, as already mentioned in §2.1. During language teaching and learning, grammar rules and vocabularies of language are taught explicitly and a lot of exercises on constructing sentences using the target rules and vocabularies are given to language learners. When the learners are able to use the target structures correctly, language learning is said to have taken place.

Krashen (2008) names the behavioural theory as the “skill-building hypothesis”. Krashen argues this theory is not a suitable language pedagogy. He claims the process of learning is too complex to be modelled with repetition tasks structured by explicit grammatical rules. He rules that linguistic experts have not come up with comprehensive grammatical rules and vocabularies of any language. As an alternative model, he proposes a new language learning theory called The Input Hypothesis.

2.2.2 The Input Hypothesis

The Input Hypothesis which is formulated by Krashen (1985), emphasises how valuable comprehensible input is, in acquiring a second language. Krashen’s suggestion is that learners just need to be exposed to the right “training data”, and they will use their natural statistical language learning rules to learn the patterns implicitly in the data. Importantly, the training data must not be too far removed from what the learners already knew. If learners understand input from listening or reading and the input is “just beyond” their current stage of language proficiency, then new learning takes place. Referring to Krashen’s formula i+1, if the learners are at a level ‘i’, then a language is
acquired when they are exposed to comprehensible input that belongs to stage ‘i + 1’. Since the primary method of getting comprehensible input is from listening or reading, it is not so important for learners to practise speaking in order to acquire language. According to Krashen, many second language learners will go through a period which is referred to as a silent period and they will begin to speak when they have received sufficient comprehensible input.

Nevertheless, Swain (1985) argues comprehensible input is insufficient for learners to acquire a second language. Swain claims language learners have to be given an opportunity to use a language they learn. Hence, she proposes the Output Hypothesis.

2.2.3 The Output Hypothesis

The Output Hypothesis says that comprehensible output as well as comprehensible input assist in language acquisition. This hypothesis is Swain’s attempt to explain how second language learners acquire a second language when they are pushed to produce output in the form of speaking and writing. Swain (2005) highlighted three functions of output in second language learning:

1. the noticing function which triggers the learners to notice their problem (do not know how to say or write), while attempting to produce a language.

2. the hypothesis-testing function which enables the learners to test the hypothesis through modifying their output during conversation or in response to feedback from others.

3. the metalinguistic or reflective function which allows the learners to produce a language in order to reflect consciously on the language produced by them or others.

Krashen (2008) claims the Output Hypothesis is implicitly similar to the Communicative approach, briefly introduced in §2.1.
2.2.4 The Communicative Approach and the Interaction Hypothesis

The Communicative language teaching and learning approach strongly encourages a communication between teachers and learners by using real-life lesson topics in teaching materials (Galloway, 1993). In this concept, rather than learn about language grammar rules, the learners learn how to use the language in a certain situation. Since the approach emphasises the requirement of communication, this approach can be related to another model called the Interaction Hypothesis.

The Interaction Hypothesis (Long, 1981) highlights the role of interaction in facilitating the acquisition of second language. It focuses on one particular type of interaction which is called “negotiation of meaning”. Negotiation of meaning is a process that is initiated when a conversation between two speakers breaks down. In language teaching and learning, the two speakers are a teacher and a language learner. Both parties are trying to solve the problem by accomplishing a variety of conversational strategies, which Long has referred to as interactional modifications (Long, 1981). The strategies are comprehension checks, confirmation checks, clarification requests, and recasts. The strategies tend to increase input comprehensibility and output modifiability, which are both helpful in acquisition of language. Comprehension checks are when a speaker asks if an addressee understands the speaker’s preceding utterance, for example “Do you understand what I mean?” Confirmation checks are the strategy used by an addressee to confirm whether he/she correctly understands the speaker’s utterance, for example (adapted from Courtney (2001):

(1) Speaker: Mexican food have a lot of ulcers?

Addressee: Mexicans have a lot of ulcers? Because of the food?

Clarification requests are used by the addressee to clarify what the speaker has just uttered, for example “I don’t understand.” or “Could you repeat?” Finally, recasts are a strategy in which the addressee repeats the speaker’s utterance by correcting the

---

3Negotiation in language classrooms is considered as negotiation of form which means by a focus on grammatical errors committed by students.
grammatical structure of the utterance. An example of this is:

(2) Speaker: Where did you go yesterday?
Addressee: I go shopping.
Speaker: Oh, you went shopping. Where?
Addressee: Yes, I went shopping at Kmart.

The interactional modifications can also be referred to feedback utterances provided by language teachers in response to language learners’ ill-formed utterances. The feedback utterances are called corrective feedback, which I will discuss further in §2.6.

Language learning through interactions gives an opportunity to learners to practise their language skills, especially the speaking skill. The use of multimedia elements and the Internet in CALL systems assist language teaching and learning to be more authentic. Examples of CALL systems which focus on interaction in language learning are online chat systems and dialogue-based CALL systems. In my thesis I will focus on dialogue-based CALL systems. In the following section I discuss a number of dialogue-based CALL systems.

2.3 Dialogue-based CALL systems

Stockwell (2007) reports statistics on CALL research trends from 2001 to 2005 which focus on language skills such as grammar, vocabulary, pronunciation, reading, writing, listening and speaking. The statistics are based on the number of articles published in each skill per year. The statistics show that writing, pronunciation and speaking skills are the skills which have caught developers’ attention over time compared to other skills. Stockwell claims the reason is because of the availability of Natural Language Processing (NLP), Automatic Speech Recognition, Text To Speech, and Computer-Mediated Communication technologies.

CMC-based CALL systems e.g. text-based or voice-based online chatting, focus on an interaction between a student and a teacher or a peer where a computer acts as a medium of communication. However, the presence of both parties is vital in these systems in order to establish the interaction. To offer opportunities for more
autonomous language learning, other CALL systems have to be developed in which the student can communicate with the computer without the presence of a teacher or a peer. Such systems are called dialogue-based CALL systems.

Dialogue-based CALL systems allow learners to engage in a natural language dialogue with a computer in order to practise their communication competence in a target language. The dialogue can be either text-based or spoken, or both. While text-based systems help to enhance reading and writing skills, speech-based systems assist with listening and speaking skills. Recently, research on dialogue-based CALL systems has become very popular (Davies, 2005), but Chapelle (2005) in her paper suggests that still more research on learner-computer interaction must be conducted.

Wik (2006) discusses about what nature of conversation a dialogue system should handle. Should the system be a scenario-based dialogue or an open-domain dialogue environment? Does the system allow limited or free input sentences? Is the system a system-initiative or a mixed-initiative? The following section describes four existing dialogue-based systems that employ one or more of these conversational types: L2tutor, SPELL, Let’s Chat, and Kaitito.

2.3.1 L2tutor

L2tutor (Price, McCalla, and Bunt, 1999) is a text-based dialogue-based CALL system developed for learning English by French students. It is a scenario-based dialogue system in which a topic of conversation is restricted to common conversations for travellers, for instance ordering meals at a restaurant. As such, its vocabulary only consists of common words for that scenario.

L2tutor also supports a mixed-initiative dialogue, which means the system and the learners can both take initiatives during a conversation. In order to interpret the learner’s input, the system uses a method called keyword to context matching. A keyword which is found in the vocabulary list, is extracted from the input. Then the keyword will be matched with the next context of the dialogue scenario during the conversation. L2tutor’s dialogue environment consists of several scenarios. For example, in the order meals at a restaurant scenario, it starts with a greeting, then
order drinks, followed by serve, order food and ends with pay bill. (See Price et al. (1999, p.92)). Suppose the current context is greeting: the extracted input keyword must match with the next context which is order drinks. A sample of dialogue between a user and L2tutor is shown in dialogue (3) below (adapted from Price et al. (1999, p. 95)):

(3)  L2tutor: Welcome to Susie's restaurant! This is a simulation of a restaurant where you can practise your English skills. At Susie's, the menu is very limited. There are, however, many more items available. At Susie's you can ask whether your favourite food is served here, and do a few simple things that are commonly done in restaurants. So feel free to experiment.
L2tutor: Howdy! Nice day, isn't it?
Learner: Yes.
L2tutor: Would you like to sit in the smoking section or the non-smoking section?
Learner: It doesn't matter to me.
L2tutor: Here is a seat in the non-smoking section for you.
L2tutor: Here’s the menu for you:
for drinks we have soft-drinks, coffee and tea
for food we have fries, pasta and steak
for dessert we have cake and ice-cream
L2tutor: Would you like something to drink?
Learner: Coffee please. Cream and sugar.
L2tutor: Okay.
L2tutor: Let me go get your order.
L2tutor: The waiter has left to get your order.
L2tutor: Here is your coffee.
L2tutor: Would you like something to eat?
Learner: How much is the steak?

Price et al. (1999) state that the key goal of L2tutor is to provide a natural and robust dialogue with learners. This means during a dialogue session, the learners
Table 2.1: Error analysis by PDA

<table>
<thead>
<tr>
<th>Error type</th>
<th>Learners’ Response</th>
<th>Error Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spelling error</td>
<td>Cofee please</td>
<td>The correct spelling of cofee is coffee</td>
</tr>
<tr>
<td>Noun followed by an adjective</td>
<td>I would like a cola large</td>
<td>The adjective ‘large’ followed the noun ‘cola’</td>
</tr>
<tr>
<td>Infinitives</td>
<td>I be fine</td>
<td>The infinitive form of the verb ‘to be’ was used</td>
</tr>
</tbody>
</table>

“practise fluency, vocabulary and grammar in a realistic setting”. L2tutor does not provide any feedback for ill-formed responses from the learners during the dialogue session. Instead, a postdialogue analysis (PDA) system is developed to analyse the ill-formed responses after the dialogue session. PDA consists of a simple parser with a limited vocabulary. Table 2.1 lists some errors together with the error messages that can be detected by PDA (adapted from Price et al. (1999, pg. 100–101)).

2.3.2 SPELL

Morton and Jack (2005) develop a speech-based CALL system which is known as Spoken Electronic Language Learning system (SPELL). SPELL incorporates recent technologies such as virtual agents, speech recognition and synthesis, and a virtual environment. SPELL is a scenario-based system similar to L2tutor. Types of scenario available are common transactions such as ordering food at a restaurant, purchasing tickets at the railway station, and chatting about family, sports and hobbies.

There are three dialogue sessions in SPELL which the learners can participate in: observational, one-to-one and interactive scenarios. The observational scenario is where the learner can watch a spoken dialogue between multiple agents given a specific scenario. The dialogue among the agents is provided by pre-recorded audio files. After watching the observational scenario, the learner can then have a one-to-one dialogue with a tutor agent. The agent asks some questions relating to the scenario the learner has just watched, as well as some questions about the learner. The agent’s dialogue is
also referred to from the pre-recorded audio files which are played depending on the conversation flow between the learner and the agent.

In the interactive scenario, the learner becomes an active participant in the virtual dialogue. Suppose a scenario is at a restaurant: the learner will virtually go with a virtual tutor to sit at a table. Then a conversation between the learner and the tutor begins. Later, a virtual waiter agent comes to take an order. Here also, the learner’s input dialogue is needed in order to make the dialogue continue.

In order to respond appropriately to learners’ utterances, SPELL has a recogniser which consists of a list of expected utterances that the learner may provide. The list is called a recognition grammar. As we know, the learner’s utterance can be well-formed and ill-formed. To detect errors in the ill-formed utterance, SPELL is equipped with a different grammar called Recognition Grammar for Language Learners (REGALL). It is a list of expected ill-formed utterances that the learner may give. In terms of responding to the learners’ ill-formed utterance, SPELL gives two types of immediate feedback: reformulation and recast. Reformulation is a feedback utterance in which an agent rephrases her/his utterance. Recast is a feedback utterance provided by the system which implicitly corrects the learner’s ungrammatical utterance.\(^4\) Dialogue (4) below is a sample of dialogue in SPELL (adapted from Morton and Jack (2005), page 185):

(4) **SPELL:** What drink does Katie like?
    Learner: [Silent]
    **SPELL:** What drink does Katie like? [Slower]
    Learner: Umm-drink ...
    **SPELL:** John likes red wine. What drink does Katie like?

2.3.3 Let’s Chat

While L2tutor and SPELL focus on specific scenarios in conversations, a dialogue-based system called Let’s Chat (Stewart and File, 2007) focuses on daily social conversations.\(^4\) Further information about these two types of feedback are described in §2.6.
The system is developed for L2 learners, either beginners or intermediate, to practise their social conversations skills. Several simple topics that the learner may select in order to converse with the system are *Friends, Food, Holidays* and *Sports*.

The difference between *Let’s Chat* and the two previous systems is in how user input is processed. Instead of implementing a keyword-matching technique or learners’ utterance templates when processing the learner’s input, *Let’s Chat* provides the learners with a set of possible responses for a question which are known as *pre-stored phrases*.

A conversational flow starts with a *topical bid*, a *topical bid reply*, an *elaboration prompt*, an *elaboration reply* and ends with a *brief story*. After the brief story, a new topic can be initiated. For every flow level, either the system or the learner may initiated a dialogue. Each learner’s selection from the pre-stored phrases will be responded to by the system. If the selection is correct, an acknowledge is praised, otherwise the learner will be alerted about her/his wrong choice. Stewart and File claim that with this type of conversation, the learners are expected to notice what is an appropriate response for a certain question or statement. From that, gradually the learners will build up confidence to engage in a conversation using the target language with real people. A sample of dialogue between an user and the system is shown in (5) (adapted from Stewart and File (2007, p. 116)):

(5) **Let’s Chat** (topical bid): Hello. What is your name?

   Learner (topical bid reply): My name is Diego.

   **Let’s Chat** (elaboration prompt): That sounds Spanish to me. Are you from Spain?

   Learner (elaboration reply): I come from Buenos Aires, the capital of Argentina.

   **Let’s Chat** (brief story): I have visited Argentina several times and spent some wonderful holidays in Buenos Aires. I love to go to the Bombonera and watch a Boca-River game.

2.3.4 **Kaitito**

The *Kaitito* system is a web-based dialogue system which is designed to teach the English and Māori languages (Vlugter, Knott, and Weatherall, 2004). The system
enables a text-based dialogue between a computer and a learner. This system supports an open-ended conversation, mixed-initiatives dialogues (Slabbers, 2005) and multi-speaker dialogues (Knott and Vlugter, 2008). The multi-speaker mode means a learner can have a conversation with several “characters” in Kaitito. The characters are represented as cartoon images.

Similar to L2tutor and SPELL, Kaitito also allows learners to enter free text. However, in Let’s Chat, the learners are bound to a response based on a list of pre-stored utterances provided. In terms of interpreting learners’ input, L2tutor processes the input by using keyword-matching techniques, and SPELL compares the learner’s sentences from a recognition grammar and generates responses from a given text-template. In Let’s Chat, there are no analyses of the user’s input, as the learners respond by selecting an utterance from a pre-stored list which consists of possible responses for posed questions or statements. On the other hand, Kaitito interprets the learner’s sentence by checking the sentence syntax using a chart parser\(^5\). The chart parser system is the Linguistic Knowledge Building (LKB) chart parser (Copestake and Flickinger, 2000). If the sentence is successfully parsed, then a semantic representation of the sentence is generated. Then Kaitito responds to the learner based on its representation of the meaning of the learner’s utterance.

The LKB system supports the English Resource Grammar (ERG) (Flickinger, 2000) as the representation of English grammar. ERG is represented in a head-driven phrase structure grammar (HPSG) formalism (Pollard and Sag, 1994). Currently the ERG of version July 2007 is used in Kaitito.

Figure 2.1 shows an architecture of Kaitito. When a user enters an input sentence, the system responds to the sentence. The process of responding is done in an interpretation and disambiguation module. In this module, firstly, the system parses the sentence using the LKB parser. The user’s sentence is parsed and generated into syntactic structures. Syntactic structures are represented in the HPSG formalism (Pollard and Sag, 1994). Each structure is associated with a semantic representation which is called Minimal Recursion Semantics (MRS) (Copestake, Flickinger, Pollard, and Sag, 2000).

\(^5\)Further information about a chart parsing approach can be referred to Jurafsky and Martin (2009) in Chapter 13.
Figure 2.1: The architecture of Kaitito
Then, the MRS form is converted into Discourse Representation Structures (DRS) (Kamp and Reyle, 1993).

A DRS can be interpreted in many ways. It can be interpreted as a question, an assertion or an answer to a question. For instance, if the previous utterance is a question, Kaitito expects the current utterance is an answer to the question. Otherwise, the current utterance is interpreted as an assertion. Based on the interpretation outcome, a dialogue manager generates a response in MRS form. Lastly, the MRS is converted to a natural sentence by a sentence generator, to respond to the user’s input.

(Prior to my work, only a small number of errors could be recognised.) Kaitito currently provides explicit corrections in response to learners’ erroneous sentences. An explicit correction is feedback that corrects the learners’ erroneous sentence by simply presenting the corrected version of the sentence. A sample dialogue in (6) shows a dialogue session between Kaitito and a learner and how Kaitito responds to an ill-formed utterance.

(6)  
Kaitito:  How are you?  
Learner:  I am fine.  
Kaitito:  Okay  
Kaitito:  What is your name?  
Learner:  My name Ann.  
Kaitito:  You said: my name * ann.  
But I think you mean:  
- my name is ann  
Please try again  
Learner:  My name is Ann.  
Kaitito:  Okay

As already mentioned before, the unique difference between Kaitito and other dialogue systems I describe here is its use of a rich sentence grammar of the target language. A CALL system that is able to check a sentence grammar or parse a sentence is known as a parser-based CALL system.

---

6Further information about explicit correction is described in §2.6.
2.4 Parser-based CALL systems

Sentence parsing is the process of assigning a grammatical structure to an input sentence, as specified by a declarative grammar. As Heift and Schulze (2007) explain, sentence parsing is one of the NLP techniques which have been used in developing CALL systems. Heift and Schulze define a CALL system that incorporates a grammar and a sentence parser as a parser-based CALL system. Parser-based CALL systems may or may not be capable of handling dialogues. Among the four dialogue-based CALL systems discussed earlier in §2.3, only the Kaitito system is considered as a parser-based CALL system. Although L2tutor has a parser to detect erroneous utterances, the parser is only activated after a dialogue session with learners is completed. In the following, I describe four existing parser-based CALL systems: German Tutor, ICICLE, Arabic ICALL, and BANZAI.

2.4.1 German Tutor

German Tutor (Heift and Nicholson, 2001) is an Intelligent Language Tutoring System (ILTS) for learning German language. The system is developed to help students practise German grammar. The German grammar formalism is represented in HPSG, similar to Kaitito. When a learner enters an input utterance, it is parsed, and the resulting grammatical representations are used to perform a series of checking mechanisms relating to missing word detection, word order and grammatical errors. If an error is found, then the system provides an error feedback utterance based on the learner’s level of language proficiency.

A learner’s language proficiency level is recorded in the learner’s profile. The profile is stored in a student model. The proficiency level is either beginner, intermediate or advanced. In order to respond to an erroneous sentence, a detailed error feedback which consists of an exact error location and the type of error, is provided to a beginner learner. For an intermediate level, only the type of error is given in the error feedback.

---

7ILTS and parser-based CALL systems are two similar things as mentioned in Heift and Schulze (2007, pg. 2).

For an advanced learner, the error feedback is a hint as to where an error is located. As an example in Heift and McFetridge (1999, pg. 60), a learner enters an ill-formed sentence as in (7) below.

(7) Sie träumt von einen Urlaub. *(She is dreaming of a vacation.)*

One of the following error feedback is provided based on the learner’s proficiency level.

1. There is a mistake with the article *einen* of the prepositional phrase.

2. There is a mistake in case with the article *einen* of the prepositional phrase.

3. This is not the correct case for the article *einen* of the prepositional phrase. *Von* assigns the dative case.

The feedback (1) is the most general and provided to expert learners. It provides a hint where an error is located (prepositional phrase) in the sentence. Intermediate learners are provided with the feedback (2) in which the feedback is more detailed than (1), providing additional information on the type of error (case). The most detailed feedback is the feedback (3), provided to beginner learners. The feedback response does not only pinpoint the location of and the type of error but also refers to the exact location of the error (dative preposition).

### 2.4.2 ICICLE

The name *ICICLE* is an acronym for “Interactive Computer Identification and Correction of Language Errors”. It is a parser-based CALL system that helps deaf students with grammatical components of their written English (Michaud, 2002). *ICICLE* also employs linguistic features such as detecting errors on sentences and generating feedback on the errors. *ICICLE* consists of two modules: the *identification module* and the *response generation module* (Michaud, McCoy, and Pennington, 2000). The identification module analyses each sentence, and if errors occur, the response generation module generates error feedback utterances.

The system’s work begins when a learner’s written text is sent to *ICICLE*, either by directly typing into the system or by loading a text file. The text is analysed for its
grammatical structure. During the analyses, a chart parser is used to parse the sentence. There are two kinds of English grammar referred to during the parsing process. First is the grammar that represents a correct structure of English sentences. This grammar is used for analysing a grammatical input. As for analysing the grammatical errors, the second grammar that consists of structure of ungrammatical sentences, known as an error grammar, is used. More about error grammars is explained in §2.7.1.1. In order to address any errors occurred, the response generation module provides feedback pertaining to the error. The feedback is generated from a list of canned explanations for the error which occurred.

2.4.3 Arabic ICALL

Arabic ICALL is also another example of a parser-based CALL system (Shaalan, 2005). The word ICALL is an abbreviation of “Intelligent Computer Assisted Language Learning”. The word “intelligent” is used because of the use of NLP features on CALL. Obviously the system is developed for learning Arabic.

Arabic ICALL consists of four components: a user interface, course materials, a sentence analyser and feedback provider. The user interface acts as a medium of communication between users and the system. The course material consists of teaching content, which includes a database of test questions, a tool to generate test scripts, and a tool to maintain lessons and test items.

The sentence analyser has a morphological analyser, a parser, grammar rules and a lexical database. The Arabic grammar is written in a definite-clause grammar (DCG) formalism. The analyser parses a learner’s sentence with the grammar. In order to handle an ungrammatical answer, similar to ICICLE, Arabic ICALL is also equipped by an Arabic error grammar. Then, the analyser passes the analysis form of the learner’s answer to feedback provider.

For each question given to a learner, its corresponding answer is provided to Arabic ICALL. The answer is also analysed by the sentence analyser in order to produce an analysis form of the answer. The feedback provider compares the analysis of learner’s answers with the analysis of the correct answer. A positive message will be issued to
the learner if it matches. Otherwise, error feedback is given based on information in the Arabic error grammar. Refer to Shaalan (2005, pg. 100–105) for further information about how learner’s ill-formed answers are handled.

2.4.4 BANZAI

BANZAI is a parser-based CALL system that helps learners to learn Japanese (Nagata, 1997). BANZAI employs a simple parser based on a finite state grammar (Gazdar and Mellish, 1989). The system is fully implemented in Lisp and described in Nagata (1997). BANZAI parses not only grammatical sentences but also a variety of erroneous sentences. The target errors are Japanese particles. When the system detects that a sentence contains errors, BANZAI diagnoses what types of errors have occurred and provides feedback. Two types of feedback are provided: deductive and inductive feedback. Besides a parsing facility, BANZAI provides a simple authoring tool which helps a teacher to provide exercises and their corresponding answers. Deductive feedback utterances give detailed linguistic information on particle errors and besides, the correct particle is also given. Inductive feedback is deductive feedback with the addition of two or three examples of correct sentences that use the particles.

2.5 Grammatical Errors in Language Learning

When learning a language, learners are certainly making many errors, especially if they are beginners. Errors are defined as occurring due to learners’ lack of competence in speaking the target language. Competence is understood in a Chomskykan sense, as knowledge (possibly implicit) of the rules of the language. Mistakes are defined as occurring due to performance errors—i.e. they result from learners’ failure to exhibit some part of the competence in the target language which they have already acquired. Mistakes arise due to lapses in concentration, demands on working memory and so on. (See the first hierarchical level of errors in Figure 2.2). I will focus on errors in Corder’s technical sense, as I am mainly interested in teaching knowledge of the target language (competence). The types of error will be discussed in §2.5.1. Their sources
Learner errors could be significant to teachers, researchers, and learners (Corder, 1967). The errors could provide the teachers with current information about the extent to which their students have grasped the target language at a particular time. Then the teachers can evaluate their teaching method and understand what language features their students still have to learn. For researchers, results of error analysis can reveal a process of how a target language is acquired; consequently researchers can propose a better methodology for language teaching and learning. For learners, their errors show how they incorrectly hypothesise a target language.

2.5.1 Types of grammatical error

In my research, I will focus on automatic error correction in a dialogue context. I only concentrate on checking grammatical errors. There are several ways of classifying such errors. One type of classification involves reference to simple surface features of errors. For instance, Dulay, Burt, and Krashen (1982) categorise errors as “omissions”, “additions”, “misinformations” and “misorderings”. Specifically:

- **Omissions** mean the absence of an item that must appear in a well-formed utterance, e.g. "*My name Maria."

- **Additions** mean the presence of an item that must not appear in a well-formed utterance, e.g. "*They like going to shopping."

- **Misinformations** mean the use of the wrong form of the morpheme or structure, e.g. "*I goed to visit my grandparents."

- **Misorderings** mean the incorrect placement of a morpheme or group of morphemes in an utterance, e.g. "*I like go to ."

Ellis (1994) notes that a surface classification like this one does not really explain the student’s error - however, it can still be useful in providing feedback to the student about what was wrong. In Chapter 4 I will describe an automated error-correction technique which uses a surface classification of errors.
There are also a large number of error taxonomies that make reference to syntactic concepts. I will discuss these in some detail in Chapter 3 (§3.3), before providing my own detailed error taxonomy tailored to the Malaysian EFL domain.

2.5.2 Sources of grammatical error

According to Taylor (1986), sources of learners’ errors may be psycholinguistic, sociolinguistic, ‘epistemic’, or may concern discourse structures. Psycholinguistic sources relate to most of what we think of as language errors, and roughly comprise Corder’s classes of competence (knowledge) and performance (processing) error. Sociolinguistic sources are related to the learners’ social communication and how they adjust their language within the context. For example, a learner may be overly formal or informal in different social situations. Epistemic error sources refer to the learners’ insufficient world knowledge. For instance, to use formal language appropriately, it is necessary to understand the structure of the society in which the target language is used. Discourse sources refer to the learners’ problems in organising information into a coherent multisentence text. It may be harder for a student to do this in the target language than in her own language, because the discourse structuring conventions may be different. Most research has focussed on psycholinguistic error sources, and I will also focus on these.

Ellis (1994) proposes three different psycholinguistic sources of error: ‘transfer’, ‘intralingual’, and ‘unique’ (see Figure 2.2). Transfer is defined by Odlin (1989) as “the influence resulting from the similarities and differences between the target language and any other language that has been previously (and perhaps imperfectly) acquired.” (Of course, “any other language” primarily includes the learners’ first or native language, L1). If a Malay-speaking ESL learner uttered “My name Maria”, this could be due to L1 transfer problem because the copula “be” does not exist in the Malay grammar. §3.8 provides some more examples of L1 transfer problems. Different studies report different incidences of transfer errors in language learners. For instance, Ellis (1985) reports percentages of transfer errors ranging from 3% to 51%. Importantly, transfer can be facilitating as well as hindering. Transfer errors are ‘negative’ transfer, where L1
patterns are wrongly incorporated into the learner’s target language, L2. But transfer can also be positive, where there are syntactic similarities between L1 and L2; in these cases, transfer is beneficial.

Not all theorists believe that L1 is an important source of students’ L2 errors. For instance, Corder (1967) disagrees that L1 is the source of errors and Dulay and Burt (1974) have provided evidence to support Corder’s claim. However, the analysis of Malay learner errors which I provide in Chapter 3 certainly appears to support the suggestion that some learner errors are due to transfer from L1.

Intralingual error sources are due to learners’ difficulties in inducing the rules of the L2 being learned from example sentences. These sources relate to faulty generalisation abilities, or faulty application of rules (or faulty recognition of exceptional cases where rules do not apply).

‘Unique’ (or ‘induced’) errors occur as a result of the instruction that an L2 learner receives. For instance, if a teacher provides a simplistic or incorrect description of some aspect of L2 syntax, this might result in particular errors by the students - especially in attentive or diligent students. In order to avoid producing induced errors in students, one important thing is to provide the right kind of feedback about the errors they make. I will turn to the topic of feedback in the next section.
2.6 Corrective Feedback

There are different ways of responding to student errors. When designing an error correction system, it is important that it provides enough information to allow a useful response. In this section, I will review research on the different types of corrective feedback which can be made by language teachers, and their effectiveness.

A piece of corrective feedback (CF) is a response from an addressee to a speaker, where the addressee’s intention is to correct the speaker’s erroneous utterance. According to Ellis et al. (2006), corrective feedback responds to learners’ erroneous utterances by i) indicating where the error has occurred, ii) providing the correct structure of the erroneous utterance, or iii) providing metalinguistic information describing the nature of the error, or any combination of these.

2.6.1 Corrective Feedback Types provided by Language Teachers

Lyster and Ranta (1997) conducted an observational study on the provision of corrective feedback by teachers at a primary school in France. They categorised the corrective feedback as being of six different types: explicit correction, recast, clarification requests, metalinguistic, elicitation and repetition. In addition, Panova and Lyster (2002) introduce two more types of feedback: translations and paralinguistic signs.

An explicit correction is a teacher’s feedback utterance in which she or he explicitly corrects a student’s erroneous utterance by providing the correct form of the utterance. For example:

(8) Student: I go to a zoo last Sunday.
Teacher: No! You should say “I went to a zoo last Sunday.”

When the teacher reformulates the student’s utterance wholly or partly in a correct form, it is called recast. For example:

(9) Teacher: What is the baby doing?
Student: The baby is cry.
Teacher: Yes, the baby is crying.
The third type of feedback is a **clarification request**. This is a teacher’s utterance which indicates that the teacher does not understand the student’s utterance or that the utterance is partly ill-formed. Therefore the student is requested to reformulate or repeat his or her utterance (Spada and Fröhlich, 1995) as cited in (Lyster and Ranta, 1997). Examples of such feedback phrases are “I don’t understand.”, “Pardon me!” or “Could you repeat that?” A sample of conversation which has this type of feedback is given below:

(10) Student: Sunday I see movie.
    Teacher: Could you repeat that?

A **metalinguistic** feedback utterance is an explanation of any errors that occurred in a student’s erroneous utterance without providing the correct answer. According to Lyster and Ranta (1997), this feedback can be either in the form of comments, information, or questions. Metalinguistic comments denote that there is an error occurring in the student’s utterance, for instance:

(11) Student: John buy some fruits.
    Teacher: No, not buy.

Metalinguistic information can be given either as a grammatical description of the ill-formed utterance or a definition of a word if there is a lexical error. An example of metalinguistic information feedback is as follows:

(12) Student: I go to a zoo last Sunday.
    Teacher: Use the past tense.

A metalinguistic question is similar to metalinguistic information, but instead of providing the information, the teacher tries to elicit it from the student. For example:

(13) Student: I go to a zoo last Sunday.
    Teacher: Past tense?

An **elicitation** feedback is the fifth type where the teacher can apply at least three methods in order to get the right utterance from the student. The first technique is by asking the student to complete the teacher’s partly utterance as shown below.
(14) Student: Tomorrow I bring the book.
    Teacher: No, tomorrow I .......... 

In the second elicitation technique, the teacher questions the student in order to 
elicit a correct utterance from the student, for instance:

(15) Student: I go to a zoo last Sunday.
    Teacher: How do we say ‘go’ in past tense?

The third technique is used when the teacher requests the student to reformulate 
his or her utterance. Here is one such instance:

(16) Student: I goed to a zoo last Sunday.
    Teacher: goed?

A repetition feedback utterance is the sixth type of feedback. The teacher repeats 
his or her student’s incorrect utterance and raises his or her voice to highlight the error 
in the utterance. An example can be as follows:

(17) Teacher: What is the baby doing?
    Student: The baby is cry.
    Teacher: The baby is cry?  [Italic font shows the increase of the teacher’s voice.]

Translation feedback is used to interpret the learner’s unsought uses of her or his 
L1 into the target language. This feedback is relatively similar to recasts and explicit 
corrections where the teacher provides the correct version of the student’s L1 utterance. 
Here the student’s L1 utterance may be a grammatical or ungrammatical form. Due 
to the student’s difficulty in producing the target language, he or she responds in L1. 
For instance:

(18) Teacher: Where did you go last Sunday?
    Student: I.. Saya pergi zoo. (L1)
    Teacher: You went to a zoo? (translation)

A paralinguistic sign is non-verbal corrective feedback where the teacher displays 
facial expression, produces gesture cues or raises her or his voice intonation in response 
to the student’s erroneous utterance. For example:
2.6.2 Explicit and Implicit Corrective Feedback

Long (1996) and Ellis et al. (2006) have identified that corrective feedback can be provided either explicitly or implicitly. Explicit corrective feedback utterance tells overtly that an error has occurred. In contrast, an implicit corrective feedback utterance does not. Kim (2004) claims there are two types of implicit CF depending on whether a correct form of an erroneous sentence is given or not. The first type (i.e. recast) provides a correct form immediately after a learner’s error. The second type (i.e. elicitation, clarification request) does not provide a correct form but encourages the learner to repair his/her erroneous sentence by asking the learner to rephrase it. Ellis, Loewen, and Erlam (2006) classify explicit corrections and metalinguistic feedback as explicit CF. Another explicit CF type is elicitation feedback (Ellis, 2007). While Ellis claims elicitation is an explicit form, Kim (2004) claims it is implicit. Ellis argues that from the provision of such feedback, a learner knows that his/her sentence has errors. On the other hand, Kim claims elicitation is the second type of implicit CF because the feedback encourages the learner to correct his/her erroneous sentence.

Referring to the previous subsection (§2.6.1), there are 3 ways that an elicitation feedback utterance can be provided. First, when a teacher asks a student to complete the teacher’s partial feedback utterance as given in Example (14). Second, when a teacher asks questions to a student in order to elicit a correct utterance from the student as shown in Example (15). Third, as given in Example (16) when a teacher requests a student to rephrase the students’ utterance. I suggest that the first two ways tend to be of an explicit form. The last one is implicit.

As suggested by Long (1996), the implicit feedback types are recast, clarification requests, repetition and paralinguistic signs. Yet, Ellis et al. (2006) and Lyster and Ranta (1997) agree that recast is an implicit CF. Translation is described by Lyster and Ranta (1997) as a subcategory of recast: effectively, it is just a recast that happens to use the learner’s native language. In this review, therefore, I will consider translation
Table 2.2: Explicit and Implicit corrective feedback

<table>
<thead>
<tr>
<th>Explicit</th>
<th>Implicit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit Correction</td>
<td>Recast</td>
</tr>
<tr>
<td>Metalinguistic</td>
<td>Repetition</td>
</tr>
<tr>
<td>Elicitation</td>
<td>Clarification Request</td>
</tr>
<tr>
<td></td>
<td>Translation</td>
</tr>
<tr>
<td></td>
<td>Paralinguistic Sign</td>
</tr>
</tbody>
</table>

to be a form of implicit CF. Table 2.2 summarises explicit and implicit CF. Other information on different classifications for CF defined by Ferreira (2006) and Ellis (2007), can be found in Appendix A.

2.6.3 Research on Corrective Feedback in Language Learning

Results from SLA studies have shown that provision of corrective feedback is beneficial to language learning (LL) (Long, 1996; Swain, 2005). Besides, Basiron, Knott, and Robins (2008) identify five actions on how CF utterances help a learner progress during LL. The actions are noticing, locating, perceiving, providing uptake and repairing. The nature of CF utterances provided to learners helps the learners to notice that their sentence is erroneous. The provision of CF also helps the learners to locate the committed error. After that, the learners perceive the CF as an error correction feedback for grammatical errors, vocabulary errors, or semantic errors. Responses from the learners immediately after the CF focusing on their previous sentence is called uptake. The response may be the learners’ intention to repair the error.

In the current section, I discuss existing studies carried out on the provision of CF during LL in three different environments: LL in normal classrooms, LL using online chatting systems and LL using CALL systems. Even though the first two settings involve interactions between humans, then use a computer as the medium of interaction. The CALL systems need only interactions between a learner and a computer. In the following subsections, I will describe some studies on CF in each learning environment.
2.6.3.1 Normal classrooms

I will discuss two research studies that are performed on classroom settings. The first study was performed by Lyster and Ranta (1997) and the second one was by Ellis, Loewen, and Erlam (2006). These two studies are selected to provide information on what is the most and least common feedback that teachers provide during language learning in classrooms, and among the different kinds of feedback which one is the most effective which helps learners progress better.

Firstly, an observational study by Lyster and Ranta (1997) analyses the use of CF provided by four teachers in French immersion classrooms at a primary school. There are 686 students which is the total number of students in four classes: 243, 146, 194, and 103 students. There are three objectives of the study. The first, is to find what types of CF and how frequently CF is provided by teachers. Results from the study reveal 6 different CF types: explicit correction, recast, clarification requests, metalinguistic, elicitation and repetition. Among all CF, recast is the most frequently used by teachers. It comprises 55% of all feedback utterances. Distributions for other CF are elicitation 14%, clarification request 11%, metalinguistic 8%, explicit correction 7%, and repetition 5%.

The second research objective is to find the types of CF which led to uptake actions by the students. A result shows the students give uptake responses to all elicitation feedback utterances (100%). The students’ uptake to other feedback such as clarification request, metalinguistic feedback, repetition, and explicit correction is 88%, 86%, 78%, and 50% respectively. The least uptake by the students is for recasts (31%).

The third research objective is to find the distribution of what combinations of CF and learners’ uptake result in “negotiation of form”. Results show metalinguistic, elicitation, clarification requests, and repetition are the feedback types which tend to result in negotiation of form. In terms of the percentage distribution, metalinguistic (46%) and elicitation (45%) are found to be the most powerful way of encouraging repairs, followed by repetition (31%) and clarification requests (27%).

---

9 An interaction between a learner and a teacher which focuses on correcting the learner’s ill-formed sentence. See also the negotiation of meaning as defined in the Interaction Hypothesis on page 13.
Ranta found out no negotiation of form occurred after the provision of recast or explicit correction. This is because both types of CF only lead the students to repeat their teacher’s correct utterance.

The second CF study is one carried out by Ellis, Loewen, and Erlam (2006). An objective of this study is to examine the effectiveness of two types of CF: metalinguistic and recast, that are provided in a normal classroom. A total of 34 participants are divided into three groups: two experimental groups (each group has 12 participants) and a control group (10 participants). Each experimental group completes two communicative tasks (30 minutes period for each task) for two consecutive days. During the communicative tasks, one group receives metalinguistic feedback and the other group receives recast in response to past tense -ed errors in a target structure. The control group does not complete the tasks and does not receive any feedback on the errors.

Testing was done in three stages: pre-test, post-test and delayed test. The pre-test is done before the instructional treatment, the post-test is conducted on the same day immediately after the treatment, and the delayed test is given 12 days later. Once pre-test scores have been taken into account, there is no significant difference between the groups in their immediate post-test scores. However, in the delayed post-test, the metalinguistic group scored significantly from the recast and control groups. Overall, Ellis et al. conclude the metalinguistic group performs better than the recast group, which answers the research question “Do learners learn more from one type of corrective feedback than from another type?”

2.6.3.2 Online chatting systems

Loewen and Erlam (2006) replicate the experiment performed by Ellis et al. (2006) to compare the effectiveness of recast and metalinguistic feedback using an online chatting system. A total of 31 participants are divided into five groups: two groups receive recast (one group has 5 and the other group has 6 participants), the other two groups receive metalinguistic feedback (6 participants in each group), and a control group (8 participants). The control group does not receive any feedback responses for their ungrammatical utterances.
The online chatting system is done in two sessions. The first session is conducted in a self-access centre which is an open room with computers. One group each from the recast and metalinguistic groups is placed there and no two students from the same chatroom are next to each other. In the second, the remaining two groups from the recast and metalinguistic groups are placed at computers in offices next to each other. Communications with teachers occur only through the online chatting system in both sessions. During online chatting, all groups except the control group complete two activities as similar to communicative tasks in Ellis et al. (2006)s’ experiment.

Testing instruments are pre-test and two post-tests. The pre-test is done two days before the treatment. The post-test 1 is conducted immediately after the treatment and the post-test 2 two weeks after the treatment. Statistical tests are conducted to examine any significant effects from each group performance and from the pre-test scores to the post-tests scores. Results yield no significant difference on the performance for the metalinguistic groups and the recast groups, between pre-test and the two post-tests. It may be that the lack of a significant effect is just due to the relatively small sample size in this experiment. (Loewen and Erlam do not compute the statistical power of their test.)

2.6.3.3 CALL systems

Here, I will discuss three research studies on CF provided in CALL systems. Heift (2004)’s is the first one, followed by Nagata (1997), and the last one is Ferreira (2006).

Heift (2004) carried out an experiment to investigate which one of three CF types is the most effective on learners’ uptake in a parser-based CALL system. The CF types are *metalinguistic*, *metalinguistic + highlighting*, and *repetition + highlighting*. The highlighting technique here is referred to elicitation feedback as given in a normal classroom. The experiment uses a CALL system known as E-Tutor which is a system to practise various exercises on German vocabulary and grammar. There are a total of 177 participants for this study. A pre-test is given to all participants, then all participants use the CALL system for 15 weeks. After that they take a post-test. During the E-Tutor usage, at least four chapters were taught. Each participant randomly receives
one type of CF utterance, that is either metalinguistic, metalinguistic + highlighting, or repetition + highlighting for each chapter. This means, all participants receive each CF type at least once. The CF is provided as responses to the participants’ incorrect responses.

Results show that the group which received metalinguistic + highlighting feedback produced the biggest number of correct responses. The group which receives metalinguistic feedback made fewer correct responses than the metalinguistic + highlighting group. A repetition + highlighting group produces the least number of correct responses amongst the other two groups. The result also shows that those that receive metalinguistic + highlighting feedback are most likely to correct their errors (87.4%) compared to those who receive metalinguistic (86.9%) and repetition + highlighting (81.7%) feedback. Pair-wise comparison tests that are performed to determine an inter-group variation show there is a significant different between metalinguistic + highlighting and repetition + highlighting, and between metalinguistic and repetition + highlighting. Even though a pre-test and a post-test are given to all participants, no test results are reported in Heift’s paper. Overall, Heift concludes that the metalinguistic + highlighting feedback is the most effective feedback at leading to learners’ uptake when learning a language using a CALL system.

Nagata (1997) conducted an experiment whose the objective was to study the effectiveness of deductive feedback and inductive feedback provided in response to Japanese particle errors, in a parser-based CALL system called BANZAI. Both types of feedback are of metalinguistic type. Deductive feedback utterances give detailed linguistic information on particle errors and besides, the correct particle is also given. Inductive feedback is also the deductive feedback, with an additional of two or three examples of correct sentences that use the particles.

A testing procedure consists of six sessions over a duration of 15 days. The testing starts with a pre-test, followed by four computer sessions, and ends with two tests (a post-test and a comprehension test). During the computer sessions, 30 participants are divided evenly into 2 groups. One group receives deductive feedback and the other
one receives inductive feedback in response to erroneous utterances.

A result of the post-test shows that the deductive-based group performs significantly better than the inductive-based group. Again here no results are yielded on statistical significance tests for a difference between the pre-test and the post-test score for each group. Overall, the deductive-based feedback provision is more effective than inductive-based feedback for language learning.

The third research is Ferreira’s study. This is also an experiment to investigate which CF is more effective between two CF strategies, that is provided when using a Web-based CALL system. Ferreira classifies the CF into two types: Giving-Answer Strategies (GAS) and Prompting-Answer Strategies (PAS). GAS consists of repetition and explicit correction and PAS includes metalinguistic and elicitation feedback respectively. The target structure learnt in the CALL system is Spanish subjunctive mood. The research question is Are PAS or GAS feedback strategies more effective for teaching the Spanish subjunctive mood for foreign language learners?

In this study there is a total of 24 participants, who are divided: two experimental groups and one control group. The experiment consists of a pre-test, treatment sessions, and a post-test. The pre-test is given to all group participants, then all groups use the CALL system during the treatment sessions. In response to incorrect answers, the first experimental group receives PAS feedback, and the second group receives GAS feedback. The control group receives only positive and negative acknowledgment in response to their answers. The sessions end with a post-test which is conducted at the end of the treatment activities.

Statistical significance tests are performed on the differences between pre- and post-test scores in the different groups, to see whether there are more gains in some groups. The gain of the PAS group was found to be significantly greater than the gain of the control group, and also (marginally) significantly greater than the gain of the GAS group (which was itself marginally significantly greater than the gain of the control group).
Table 2.3: The tractability of different kinds of CF.

<table>
<thead>
<tr>
<th></th>
<th>Notice</th>
<th>Locate</th>
<th>Perceive</th>
<th>Uptake</th>
<th>Repair</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Elicitation</td>
<td>directly</td>
<td>directly</td>
<td>easy</td>
<td>yes</td>
<td>easy</td>
</tr>
<tr>
<td>2 Metalinguistic</td>
<td>directly</td>
<td>directly</td>
<td>easy</td>
<td>yes</td>
<td>easy</td>
</tr>
<tr>
<td>3 Explicit Correction</td>
<td>directly</td>
<td>directly</td>
<td>easy</td>
<td>least</td>
<td>easy</td>
</tr>
<tr>
<td>4 Repetition</td>
<td>directly</td>
<td>directly</td>
<td>difficult</td>
<td>yes</td>
<td>easy</td>
</tr>
<tr>
<td>5 Clarification Request</td>
<td>directly</td>
<td>indirectly</td>
<td>difficult</td>
<td>yes</td>
<td>difficult</td>
</tr>
<tr>
<td>6 Recast</td>
<td>indirectly</td>
<td>indirectly</td>
<td>difficult</td>
<td>least</td>
<td>easy</td>
</tr>
</tbody>
</table>

2.6.4 Discussion

SLA research results have proven that provision of corrective feedback is beneficial to language learning (LL) as agreed by Long (1996); Swain (2005). A provision of CF utterances leads to five actions performed by learners that can assist them in LL (Basiron et al., 2008). The actions are noticing, locating, perceiving, providing uptake, and repairing. However, not all types of CF can stimulate the learners to perform all actions. Table 2.3 summarises how CF utterances\(^\text{11}\) help the learners progress in LL according to five actions. The CF types are arranged from easy to difficult levels performing the action. Referring to Table 2.3, metalinguistic and elicitation feedback are the feedback that help learners to notice directly and locate their committed error, easily perceive the error, easily uptake/response to the error and easily repair the error. Recast is the least effective feedback that help learners to carry out those actions.

Among all different types of CF, not all are suitably used in all language learning environments. In Lyster and Ranta (1997), despite the fact that recast is the most frequent feedback provided by teachers, it makes the least contribution to the learners’ uptake and form negotiating. Although the teachers do not provide elicitation and metalinguistic feedback as much as recast, these types of CF are proved to be the two most effective feedback that help the learners during LL (Lyster and Ranta, 1997; Ellis

\(^{11}\)Only CF identified by Lyster and Ranta (1997) are included.
et al., 2006). Research performed by Ellis et al. (2006) (in a classroom), and Heift (2004); Ferreira (2006) (using a CALL system) reveal that metalinguistic feedback is the most effective feedback that help the learners during LL. In contrast, results from Loewen and Erlam (2006) (an online chatting system) show no significant increase between pre- and post-test scores at all for metalinguistic and recast groups, either across groups or within groups. (Again, no statistical power analysis is provided, so it is hard to assess this null result.)

As regards the studies discussed before have shown that between recast and metalinguistic feedback, metalinguistic feedback is the most effective feedback. This means, with a provision of metalinguistic feedback utterances, the feedback utterance assists the learners to become aware of errors they committed and correct the errors. However, in Ellis et al. (2006); Heift (2004); Nagata (1997), the effectiveness is measured on a difference of post-test score between two experimental groups. No reports of significance for a difference between a pre-test and a post-test score have been given. Only Ferreira (2006) reports a statistically significant test result that language learners perform significantly better after being provided with Prompting-Answer Strategies in which metalinguistic feedback is included in feedback strategy.

Referring back to the findings from Lyster and Ranta (1997)’s observational study, a question can be asked. Why do language teachers prefer to provide recast than metalinguistic even though research have shown that metalinguistic is more effective than recast? Ellis (2005) claims providing metalinguistic feedback discourages learners from constructing new sentences in LL. On the other hand, provision of recasts have proved to help learners who are beginners, not advanced learners (MacKey and Philp, 1998) as in Ellis (1999). Furthermore, Lyster and Ranta (1997) claim teachers provide less amount of recast feedback to students whose have higher proficiency level. These reasons also strengthen a quote from Havranek and Cesnik (2001) as cited in (Heift, 2004): “the success of corrective feedback is affected by its format, the type of error, and certain learner characteristics such as verbal intelligence, level of proficiency, and the learner’s attitude toward correction”.

41
2.7 Survey of Automatic Error Correction Techniques

Error correction for an erroneous utterance is a process in which the utterance is reconstructed as an error-free utterance. According to Sun et al. (2007), there are two basic techniques for error correction. One employs symbolic grammar-based and the other one implements statistical-based methods. I first explain the symbolic grammar error correction in §2.7.1 and then the statistical techniques in §2.7.2.

2.7.1 Grammar-based Error Correction

Parsing a sentence is the process of examining the syntax of the sentence by assigning a word category to each word, and providing phrase labels to corresponding word sequences that make up the sentence (Jurafsky and Martin, 2009). Table 2.4 and Table 2.5 list some examples of word categories and phrase labels for the English language.

Each language has its own grammar, that consists of a list of rules of how well-formed sentences are constructed. For example, to parse an English sentence, we need to specify the English grammar. In NLP, the output of sentence parsing is a representation of the structure of the sentence. In this representation, each word in the sentence is associated with its corresponding word category, and the way these words combine hierarchically grammatical constituents or phrases is shown. A graphical way to show the representation is in a form of parse tree. Figure 2.3 depicts an example of a parse tree for the sentence “The boy chased the girl.”

The parse tree of a sentence also contains information about how the sentence is interpreted. For instance, the sentence in (20) below

(20) The man fed her fish food.

has more than one interpretation as shown in Figure 2.4. The parse tree in (2.4a) is interpreted as There is a man who fed a fish that belongs to a lady or a girl with some food. Another interpretation is There is man who fed someone else with fish food as in (2.4b). Sentences that have more than one interpretation or meaning are called ambiguous sentences.
Table 2.4: Word categories for English.

<table>
<thead>
<tr>
<th>Word Categories</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjective (Adj)</td>
<td>pretty, nice, good</td>
</tr>
<tr>
<td>Adverb (Adv)</td>
<td>slowly, carefully</td>
</tr>
<tr>
<td>Determiner (Det)</td>
<td>a, an, the</td>
</tr>
<tr>
<td>Noun</td>
<td>boy, fish, cat</td>
</tr>
<tr>
<td>Possessive Pronoun (Poss-Pron)</td>
<td>her, his, your</td>
</tr>
<tr>
<td>Proper Noun (PN)</td>
<td>John, Mary, Malaysia</td>
</tr>
<tr>
<td>Preposition (Prep)</td>
<td>to, with, by</td>
</tr>
<tr>
<td>Pronoun (Pron)</td>
<td>my, he, she, you</td>
</tr>
<tr>
<td>Verb</td>
<td>eat, drink, sleep</td>
</tr>
</tbody>
</table>

Table 2.5: Phrase labels for English.

<table>
<thead>
<tr>
<th>Phrase Labels</th>
<th>Examples of Word Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noun Phrase (NP)</td>
<td>Det Noun, Pron, Det Adj Noun</td>
</tr>
<tr>
<td>Preposition Phrase (PP)</td>
<td>Prep NP</td>
</tr>
<tr>
<td>Verb Phrase (VP)</td>
<td>Verb, Verb NP</td>
</tr>
<tr>
<td>Sentence (S)</td>
<td>NP VP</td>
</tr>
</tbody>
</table>

Figure 2.3: A parse tree for “The boy chased the girl.”
A simple way to computationally present a grammar is in a context free grammar (CFG) representation. A CFG consists of a set of rewriting rules to define a formal language. Table 2.6 shows an example of CFG rules for a small fragment of English. Each rewrite rule has a left-hand side (LHS) and a right-hand side (RHS). The former is located before the arrow and the latter is located after the arrow. The LHS consists of a non-terminal symbol and the RHS may have a non-terminal or terminal symbol/s. A non-terminal symbol means the symbol can be rewritten in a different rule. On the other hand, a terminal symbol can’t be rewritten anymore. For example, in the grammar in Table 2.6 the symbols S, NP, VP, Det, Noun, Verb are the non-terminal symbols, and the symbols *the*, *boy*, *girl*, and *chased* are the terminal symbols. An interpretation for CFG starts with the first rule, Rule 1. The rule says “A sentence, *S* can be rewritten as a noun phrase, *NP* followed by a verb phrase, *VP*”. Then Rule 2 says “The noun phrase *NP* can be decomposed to a determiner, *Det* then a noun, *Noun*”. The interpretation goes on until rules which have terminal symbols on their RHS are fired.

The CFG formalism is the simplest form in which to represent a basic grammar. As we know, a natural language is complex and its sentences can be complicated but grammatical as long as they follow its rules of grammatical structure. The English language has some features to be considered when creating sentences, such as word inflections (i.e. *sing, sings, singing*,) singular and plural forms (i.e. *mug* and *mugs*), and agreement between subjects and verbs (i.e. *he sleeps* and *they sleep*).
### Table 2.6: Context free grammar rules

<table>
<thead>
<tr>
<th>Rule 1:</th>
<th>Rule 5:</th>
</tr>
</thead>
<tbody>
<tr>
<td>S → NP VP</td>
<td>Noun → boy</td>
</tr>
<tr>
<td>Rule 2:</td>
<td>Rule 6:</td>
</tr>
<tr>
<td>NP → Det Noun</td>
<td>Noun → girl</td>
</tr>
<tr>
<td>Rule 3:</td>
<td>Rule 7:</td>
</tr>
<tr>
<td>VP → Verb NP</td>
<td>Verb → chased</td>
</tr>
<tr>
<td>Rule 4:</td>
<td></td>
</tr>
<tr>
<td>Det → the</td>
<td></td>
</tr>
</tbody>
</table>

#### 2.7.1.1 Error Grammars

Error detection and correction are two crucial components of CALL systems. Error detection is a process of identifying (and perhaps representing) any errors in a sentence. Error correction is the process of correcting any such errors. As argued by Menzel and Schröder (2003), in order for a CALL system to support effective interaction with language learners, it must possess two features. The first one is to have a robust parser and the second feature is an ability to perform error diagnosis. A robust parser is a parser that can parse a sentence even if it contains some errors. Error diagnosis is the process of identifying errors explicitly in order to correct the errors. An outcome of error diagnosis should be presented informatively to the language learners in which they are able to understand what errors they committed and how can they correct the errors.

There are several approaches used in developing robust parsers. One approach is to explicitly include grammar rules which model particular classes of error. Another approach is to implement grammars in a way that allows grammatical constraints to be relaxed. For error grammar rules, besides having grammar rules which represent well-formed structures of sentences, grammar rules that represent the structure of ill-formed sentences are also provided. These rules are also called error rules or mal-rules. Table 2.7 shows a small list of grammar rules in which condition of singular (s) and plural (p) features are added. Rule 1 is interpreted as a sentence, S is decomposed to a singular noun phrase, NP-s followed by a singular verb phrase, VP-s. Among all rules, only Rule 10 represents a mal-rule. An asterisk symbol (*) is used to indicate a mal-rule. Clearly, Rule 10 violates the subject-verb agreement. During parsing, if this rule is fired, we know that an ungrammatical sentence is parsed, and we know
Table 2.7: (Error) Grammar rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1:</td>
<td>( S \rightarrow NP-s \ VP-s )</td>
</tr>
<tr>
<td>Rule 2:</td>
<td>( NP-s \rightarrow Det-s \ Noun-s )</td>
</tr>
<tr>
<td>Rule 3:</td>
<td>( V-s \rightarrow Verb-s \ NP-s )</td>
</tr>
<tr>
<td>Rule 4:</td>
<td>( Det-s \rightarrow a )</td>
</tr>
<tr>
<td>Rule 5:</td>
<td>( Noun-s \rightarrow boy )</td>
</tr>
<tr>
<td>Rule 6:</td>
<td>( Noun-s \rightarrow girl )</td>
</tr>
<tr>
<td>Rule 7:</td>
<td>( Verb-s \rightarrow chases )</td>
</tr>
<tr>
<td>Rule 8:</td>
<td>( Verb-p \rightarrow chase )</td>
</tr>
<tr>
<td>Rule 9:</td>
<td>( VP-p \rightarrow Verb-p \ NP-s )</td>
</tr>
<tr>
<td>Rule 10:</td>
<td>( S* \rightarrow NP-s \ VP-p )</td>
</tr>
</tbody>
</table>

exactly what kind of error has been made.

Bender, Flickinger, Oepen, Walsh, and Baldwin (2004) develop a small system which has mal-rules that can be used in CALL. The system is called Arboretum. A grammar representation in Arboretum is actually the English Resource Grammar (ERG) augmented with some mal-rules. When an ungrammatical sentence is parsed using the mal-rules, its correct meaning is produced. Therefore from the meaning, a corresponding well-formed sentence structure can be generated. Yet from the fired mal-rules, analysis of errors can be done. For example an ill-formed sentence “the mouse run” is parsed by Arboretum and its parse tree is depicted in Figure (2.5a). The sentence is then corrected to “the mouse runs” as in Figure (2.5b).

Another system that implements error grammars is described by Michaud (2002)’s CALL system, ICICLE which has already mentioned in §2.4.2. The error grammar rules work fine with simple errors but become hard to use when a large grammar is involved. As mentioned in Prost (2009), the main drawback of this approach is when parsing unexpected sentences. The idea of mal-rules is basically trying to forecast every erroneous sentence that language learners could make. It is impossible to anticipate every single error rule the language learners may use because the range of errors is very big.

2.7.1.2 Constraint Relaxation

Instead of providing the error rules, as an alternative, conditions of some well-formed sentence structure are ignored temporarily until a consistent sentence structure is found. This technique is called a constraint relaxation method.
a) An ill-formed version.

b) A corrected version.

Figure 2.5: A parse tree for mal-rules in (a) and correct rules in (b).
In contrast with the error grammar technique, when parsing an ill-formed sentence using the constraint relaxation approach, conditions for grammar rules are temporarily ignored until the sentence is successfully parsed. An example of the constraint is agreement between subject and verb, or the constraint that singular count nouns must have a determiner.

In a rule-based system called The Editor's Assistant, Douglas and Dale (1992) apply constraint relaxation rules on the grammar representation of the system. The grammar is PATR-II: a grammar formalism similar to CFG rules augmented with a set of features (Wikipedia, 2010). Each grammar rule is accompanied by constraint rules. When an input sentence can’t be parsed, constraints are removed one by one until parsing is accomplished.

I will give an example adapted from Douglas and Dale (1992). Table 2.8 outlines a small sample of PATR-II rules together with constraint relaxations. There are four constraint rules for \( X_0 \to X_1 X_2 \). A constraint rule 4 says there must be an agreement between X1 and X2. Suppose a NP phrase is “this houses”, the word “this” is a singular determiner and the word “houses” is a plural noun. Obviously, the phrase violates the constraint rule 4, so this rule must be removed. Once removed, the phrase now can be parsed with the \( X_0 \to X_1 X_2 \) rule. Nevertheless, the removal of the constraint rule is not permanent, because when a parse tree is produced, all relaxed constraint rules are extracted back for error diagnosis.

A disadvantage of the constraint relaxation approach is a parsing process could fail if no rules are fired. Therefore, to prevent such an unsuccessful parsing problem, Fouvry (2003) assigns weight values to typed feature structures of defined grammar.
rules, and Foth, Menzel, and Schröder (2005) assign values to defined constraint rules. Although these two approaches provide a good support for error diagnosis, they suffer from an efficiency of search space problem (Menzel and Schröder, 2003).

In addition, Menzel and Schröder claim the constraint relaxation approach must rely on a very strong foundation of grammar structure. This dependency leads this approach to be suitable for domain restricted applications. Yet, not all types of error can be detected. Constraint relaxation is suitable for correcting word misuse such as inflected verbs in subject-verb agreement or an incorrect determiner. Prost (2009) states “…error patterns such as word order, co-occurrence, uniqueness, mutual exclusion, …cannot be tackled.”

Sentence ambiguity is also a problem for any error correction techniques which rely on symbolic parsing. How can one identify the most appropriate parse tree among all parse trees generated? This problem is not addressed in constraint relaxation systems or in error grammar approaches. I will describe statistical error correction techniques, which to some extent address the problem.

2.7.2 Statistical Error Correction

This section explores existing research on statistical techniques in error correction. Two approaches will be discussed. The first one is statistical grammars, which I explain in §2.7.2.1. The other approach makes use of other AI techniques such as machine learning, machine translation and language modelling (§2.7.2.2).

2.7.2.1 Statistical Grammar

A statistical grammar is a regular symbolic grammar in which each rule is assigned a probability value. The value is estimated from a corpus of hand-parsed sentences. Parsing a sentence using a statistical grammar is called statistical parsing. Statistical parsers became popular in grammar-based error correction systems because of the availability of hand-parsed corpora.

Charniak (1997, pg. 9) defines a statistical parser as “a parser that works by assigning probabilities to possible parses of a sentence, locating the most probable
Table 2.9: Probabilistic context free grammar rules with a probability value (in bracket) assigned to each rule

<table>
<thead>
<tr>
<th>Rule</th>
<th>Left-hand side</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S → NP VP</td>
<td>(1.0)</td>
</tr>
<tr>
<td>2</td>
<td>NP → Noun</td>
<td>(0.3)</td>
</tr>
<tr>
<td>3</td>
<td>NP → Pron</td>
<td>(0.3)</td>
</tr>
<tr>
<td>4</td>
<td>NP → Poss-Pron Noun</td>
<td>(0.3)</td>
</tr>
<tr>
<td>5</td>
<td>NP → NP NP</td>
<td>(0.1)</td>
</tr>
<tr>
<td>6</td>
<td>VP → Verb NP</td>
<td>(0.7)</td>
</tr>
<tr>
<td>7</td>
<td>VP → Verb NP NP</td>
<td>(0.3)</td>
</tr>
<tr>
<td>8</td>
<td>Noun → fish</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Noun → food</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Pron → he</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Pron → her</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Poss-Pron → her</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Verb → fed</td>
<td></td>
</tr>
</tbody>
</table>

parse tree, and then presenting that parse as the answer”. In order to estimate the probabilities, a corpus of handparsed sentences is required. The most probable parse is a parse tree with the highest probability value.

Bod (2008) claims a typical statistical parser uses a predefined grammar with a probability value assigned to each grammar rule. A probabilistic context free grammar (PCFG) is one of the simplest examples of a probabilistic grammar formalism. Each rule is assigned a probability. If there is more than one rule for a non-terminal symbol, the sum of the probability value for all those rules must be one. For instance, as shown in Table 2.9, there are 4 NP rules (Rule 2, 3, 4, and 5), so the sum of probabilities for those rules is one. For illustrative purposes, a zero probability is assigned to each part-of-speech (POS) rule (Rule 8 to 13). An example is provided in Figure 2.6. Suppose a sentence “He fed her fish food.” which has two parse trees.

The probability of a parse tree is calculated by multiplying the probabilities of each fired rule. For an instance, in parse tree (2.6a), all rules except Rule 5, Rule 6, Rule 11 are fired. Therefore the parse tree (2.6a) has the probability \(1.0 \times 0.3 \times 0.3 \times 0.3 \times 0.3 = 0.008\). Therefore the parse tree (2.6b) has the probability \(1.0 \times 0.3 \times 0.7 \times 0.1 \times 0.3 \times 0.1 \times 0.3 \times 0.3 = 0.00006\). The parse tree (2.6a) is considered the most appropriate parse tree because its value is higher than the other one.

Bod (2008) presents another technique of statistical parsing known as data-oriented parsing (DOP). In DOP systems, estimation of probabilities is based directly on a
corpus of hand-parsed sentences. Some systems use a small subset of the corpus as a grammar (Charniak, 1997; Collins, 1999); other use the whole corpus as a grammar, with some restrictions on parsing, for example an estimation of probability based on head-words of phrases in parse trees (Collins, 1996) or adding contextual information on higher nodes in parse trees (Johnson, 1998). Refer to Collins (2003); Bod (2008) for various models of statistical parsing. Readers may refer to Jurafsky and Martin (2009); Manning and Schütze (1999) for more overview about statistical grammar.

### 2.7.2.2 Statistical Techniques in Error Correction

According to Charniak’s paper, more research is needed for statistical parsers which produce parse trees that do not exist in the training corpus. Charniak suggests “...eventually we must move beyond tree-bank styles.” Apart from statistical parsing techniques which make use of tree-bank corpora, other statistical methods have been applied in correcting errors.

The basic idea in statistical language modelling is to treat a sentence as a sequence of words, whose probability is influenced by a number of different features. Features can be other words in the sentence, or other surface characteristics of these words,
for instance their position, or their part of speech. Machine learning is a technique that learns the features extracted from empirical data or a corpus to improve the performance of a system. By performing these methods, whether in conjunction with a parser or separately, a corpus becomes a very valuable resource.

Next I present some existing research studies performed which employ statistical methods in error detection and error correction. Chodorow, Tetreault, and Han (2007) develop a system to detect preposition errors in non-native English sentences. The types of preposition errors are: incorrect choice of prepositions, unnecessary use of prepositions, and omission of prepositions. A maximum entropy model (ME) is employed to estimate a probability of 34 prepositions, based on some local contextual features. Examples of the features extracted from the corpora are preceding noun (PN), preceding verb (PV), trigram to left; includes two preceding words and POS (TGL), and trigram to right; includes two following words and POS (TGR).12

The ME model is trained on two corpora (Lexile text and newspaper text from San Jose Mercury News), annotated with 25 contextual features. The outcome of ME training is a list of each feature with its corresponding frequency score. The model was evaluated on test data that consists of randomly selected English as a second language (ESL) writing essays by Chinese, Japanese and Russian native speakers. The evaluation result (precision/recall) shows that 80%/30% is achieved for detecting preposition errors.

Sun, Liu, Cong, Zhou, Xiong, Lee, and Lin (2007) developed a machine learning system to automatically identify whether a sentence is correct or erroneous. The system employs a combination of pattern discovery and supervised learning techniques. A training corpus consists of grammatical and ill-formed sentences. For each sentence in the corpus, each POS of the sentence is annotated by using the MXPOST-Maximum Entropy POS toolkit. Only function words13 and time words14 are not annotated. For instance, the sentence “John went to the library yesterday” is converted to “NNP VBD to the NN yesterday” where NNP, VBD and NN are POS tags.

12Refer to Table 1 in Chodorow et al. (2007, pg. 28) for some features used in the ME model.
13Function words are determiners, pronouns, auxiliaries, prepositions, conjunctions and adverbs.
14Time words are words about time i.e. afternoon, ago, before, and passed.
To develop a machine learning model, labelled sequential patterns (LSP) are generated from the training corpus using an existing sequential pattern mining algorithm. An LSP is represented in the (LHS → C) form, where LHS is a list of symbols and C is a class label. An example of LSP for an erroneous sentence is \(<a, \text{NNS}> \rightarrow \text{Error}\) which means a singular determiner \(a\) precedes a plural noun (NNS). An example of LSP for a correct sentence is \(<\text{would}, \text{VB}> \rightarrow \text{Correct}\), this means the word \text{would} must be followed by a verb VB. Each LSP has two scale values: support and confidence. The support value indicates the percentage of the LSP frequency in the database. The confidence value shows the higher value is the higher chance LSP can detect correct or erroneous sentences.\(^{15}\) Other linguistic features are also provided to LSP such as lexical collocation, perplexity values from language model, syntactic score and function word density.

Sun et al. evaluate their system using two test datasets: Japanese corpus (JC) and Chinese corpus (CC). These corpora consist of grammatical and ungrammatical English sentences produced by Japanese and Chinese learners respectively. The evaluation results show the highest accuracy of the system is 82%. The system is also compared with other systems: the grammar checker of Microsoft Word 03 and ALEK\(^{16}\) (Chodorow and Leacock, 2000). Sun et al.’s system outperforms the two systems in its accuracy, precision and recall scores. However, the system only detects errors and doesn’t provide suggested corrections to an erroneous sentence.

In the following, I will discuss two error correction systems that can detect and correct errors. First a system developed by Gamon, Gao, Brockett, and Klementiev (2008); Gamon, Claudia Leacock, Dolan, Gao, Belenko, and Klementiev (2009) to detect and correct errors committed by ESL writers. Contextual spelling correction and language modelling techniques are applied to the system. While the contextual spelling correction checks a word’s spelling which must be relevant to its context (i.e. Golding, Roth, Mooney, and Cardie (1999)) Gamon et al. (2008) use this technique to check whether a word is used appropriately in a given context. The system deals

\(^{15}\)See Sun et al. (2007, pg. 83–84) for further details about LSP.

\(^{16}\)A system to detect English grammatical errors that employs an unsupervised learning method.
with 8 different types of errors: prepositions, determiners, gerunds, auxiliary verbs, infected verbs, adjectives, word order and mass nouns. However, Gamon et al. report the performance of correcting determiners (a/an and the) and prepositions (about, as, at, by, for, from, in, of, on, since, to, with) errors only.

The system consists of 3 components: a suggestion provider, a language model, and an example provider. The suggestion provider (SP) component consists of the eight modules of targeted errors. In each module of the SP, machine learning and heuristic methods are employed to suggest corrections for an ill-formed sentence. Since the researchers do not have a corpus that consists of a list of pairs of ungrammatical sentences and their corrected versions, they train the machine learning framework on the English Encarta encyclopedia and a set of 1 million sentences randomly extracted from the Reuters news corpus.

In the training process, for each determiner or preposition that occurred in the corpus, context features are extracted for each of six words located to the left and right of the determiner/preposition. The features are the word’s relative position, its word string, and its POS. Then two decision trees are employed. The first decision tree is to classify whether or not a determiner/preposition should be present (pa classifier). The second one is to suggest which one determiner/preposition is the most likely preferred, provided that a determiner/preposition is present (ch classifier). The classifiers assign a probability score for each determiner/preposition value, and the highest score is selected as a suggestion.

If a sentence has an error of missing a determiner, then SP will suggest corrections which have a determiner in the corrected version. For example, consider the following ill-formed sentence:

He is lecturer from Malaysia.

The sentence is tokenised and annotated as follows:

0/He/PRP 1/is/VBP 2/lecturer/NN 3/from/IN 4/Malaysia/NNP 5/./. .

SP figures out there is a possibility that a noun phrase (lecturer) could be preceded with a determiner (a determiner + lecturer). The pa classifier assigns the probability,
e.g. \( p(\text{a determiner + lecturer}) = 0.6 \). Assume the probability of the presence of a determiner is higher than the probability of an absence of a determiner, the \( ch \) classifier is applied to suggest what is the most likely choice of determiner. Suppose the \( ch \) classifier assigns the probability of the most likely choice of determiner as below:

\[
\begin{align*}
    p(a/an) &= 0.9 \\
    p(\text{the}) &= 0.5
\end{align*}
\]

From the scores, the highest one is selected, so the SP module suggests the candidate correction for He is lecturer from Malaysia is He is a lecturer from Malaysia.

Once the correction candidate is suggested, it is passed to the second component which is a language model (LM). The LM module uses Kneser-Ney smoothing (Kneser and Ney, 1995) to estimate a probability for each candidate, as well as the input sentence. The model is trained on the English Gigaword corpus. If the score of an input sentence is less than the score of its candidate corrections, the corrections are suggested to learners.

The last component is an example provider (EP), which is an optional tool that can be activated by the learners if they need more examples for the suggested corrections. The examples are retrieved from the World Wide Web (WWW). Gamon et al. (2008) claim learners are supposed to choose the most appropriate one from the examples or just to obtain information which helps them to learn right wording for a given context. An evaluation is done separately for the SP and LM components. Gamon et al. report they achieve above 80% accuracy for both determiners and preposition corrections. However in LM, the accuracy for preposition corrections is 67%, and for determiner corrections, 51%.

Next is Lee (2009)’s work to develop a grammar error correction module for a conversational system configured for a specific domain. Four POS errors are targeted: determiners, prepositions (10 selected prepositions), verbs and noun numbers. Two phases are involved in correcting errors: overgenerate and rerank. Firstly, in the overgenerate phase, an input sentence is converted to an “over-generated word lattice” form. This form represents the sentence with all its determiners, prepositions and auxiliaries removed. All nouns are converted to their singular form and verbs are reduced
to their root word. From the lattice, the determiners, prepositions, and auxiliaries are inserted at every position in the reduced sentence. As for the nouns and verbs, their inflections are also replaced in sequence.

In the second phase, language models are implemented to score and rerank candidates in the lattice produced from the overgenerate phase. A word trigram language model and three types of stochastic CFG models are used during the reranking process. The first reranking is the trigram language model which produces a list of candidate corrections based on the lattice. From the top 10 in the list, each candidate is parsed using the three grammar models. The grammar models are: \textsc{parse}_{\text{generic}}, \textsc{parse}_{\text{generic-geo}}, and \textsc{parse}_{\text{domain}}, which is classified from more open domain to more specific domain. 

See Stephanie (1992) for more details about the grammar models. The candidate that has the highest parsing score in the top 10 list is selected. If the candidate can not be parsed, the highest score in the trigram model is taken by default.

An evaluation is performed on noun/verb and auxiliary/preposition/determiner error classes. The precision and recall for the insertion of determiners and prepositions are calculated for the four language models. The accuracy is calculated in predicting nouns and verbs for the four language models. The results show the precision and recall have improved when reranking with more specific grammar model. The best result is when reranking using \textsc{parse}_{\text{domain}} with 83% precision and 73% recall. The highest accuracy is 91% which is also when using \textsc{parse}_{\text{domain}}. This shows that the parsing performance using the three grammar models reranking strategy, is significantly better than parsing using the trigram language model alone. However, performance differences between the three grammar models are not statistically significant.\footnote{Refer to Chapter 8 in Lee (2009, pg. 75–90) for further information about the evaluation results.}

\textbf{Statistical Machine Translation based error correction}

In this section, I will describe a very interesting approach to error correction, which is based on using machine translation techniques. An influential study in this area is that of Brockett, Dolan, and Gamon (2006). The system implements phrasal Statistical Machine Translation (SMT) (Brown, Pietra, Pietra, and Mercer, 1993) techniques to
correct mass/count noun errors in learners’ ESL writing. The insight of Brockett et al. is that correction is very much like translation. For instance, consider a situation where an EFL student says “And I learned many informations about Christmas while I was preparing this article”, and her teacher corrects this to “And I learned a lot about Christmas while I was preparing this article”. The teacher is a little like a translator, who translates the students’ error-prone English into a different language, ‘correct English’. If we gather a large corpus of corrections of this kind, we can train a model which maps incorrect strings in learner sentences to correct strings in English. This can be done using a standard statistical MT paradigm called the noisy channel model.

The noisy channel model is based on the Bayes rule. The model has been applied to various areas e.g. speech recognition, optical character recognition, MT and POS tagging (Manning and Schütze, 1999). SMT is an application of Bayes’ rule and is used to show that the probability that a string of words is a translation of a source string is proportional to the product of the probability that a source string is a translation of target string (Dorr, Jordan, and Benoit, 1998). In SMT-based error correction systems, to search for an optimal correct sentence $T^*$ given an ESL ill-formed sentence $S$, the highest value of the product of the probability of $T$ and the probability of $S$ given $T$ is computed. The calculation is the SMT formula which is shown in Equation (2.1)

\[
T^* = \arg \max_T \{ P(T|S) \} \tag{2.1}
\]

\[
= \arg \max_T \{ P(T) * P(S|T) \}
\]

$P(S|T)$ is calculated from a corpus of sentence corrections. Each sentence correction is a pair of sentences, in which the first item is a learner sentence and the second item is a corrected version of this sentence. (This corpus is analogous to an aligned corpus of translated sentences in conventional SMT.) $P(T)$ is calculated from a corpus of Reuters Limited articles ($> 484$ million words), released between 1995 and 1998, and a collection of articles from multiple news sources from 2004-2005 ($> 7$K words).

Brockett et al. created their sentence correction corpus using automated string-rewriting techniques, rather than actual corrections of actual learner language. The

\footnote{The formula is taken directly from Brockett et al. (2006, pg. 250).}
creation of phrasal translation collections firstly starts with preparing a list of English nouns that are frequently involved in mass/count noun errors in the Chinese ESL learners’ writings, and searching for the intersection of words from two sources. The first source is from either the Longman Dictionary of Contemporary English or the American Heritage Dictionary. The second source is from the Chinese Learner Error Corpus (CLEC) in which the noun word occurred is tagged either as mass/count error or else with article error tag. There are 14 noun words identified: *knowledge, food, homework, fruit, news, color, nutrition, equipment, paper, advice, haste, information, lunch, and tea.* These noun word errors occurred in 46 sentences in CLEC. The translation is created by manipulating well-formed English sentences from the newswire / newspaper corpora mentioned previously. From these two corpora, 24,000 sentences containing examples of 14 targeted mass or count noun constructions are extracted, and sentence correction pairs are created by applying regular expression transformations, to create 65,000 items of the form `<learner-sentence>, <corrected-sentence>`. Examples of items include `<I’m not trying to give you a legal advice>, <I’m not trying to give you legal advice>`, or `<I’m not trying to give you the legal advices>, <I’m not trying to give you legal advice>`. 24,000 additional items were added in which a correct newswire sentence is mapped to itself (so the model does not learn that corrections are always needed).

Brockett et al. test their SMT-based model on 123 sentences including the targeted mass/count nouns, taken from English language websites in China. They report that the SMT system is able to successfully correct 61.81% of errors where corrections needed to be made.

### 2.7.3 Discussion

The task of detecting and correcting grammar errors has been a focus of research in CALL since NLP techniques were introduced to CALL systems. In order to correct language learner errors while accessing a CALL system, the CALL system requires a parser that is not only able to detect the errors, but at the same time able to provide information on how to locate and/or correct the errors. Also, the information must be
easily comprehended by the language learners.

From my observation of reviewing the existing studies, three issues are highlighted. Firstly, *types of error targeted* in error correction. Some error correction systems target certain types of POS errors. Chodorow *et al.* (2007) concentrates on detecting preposition errors only. A targeted error coverage in Lee (2009) is limited to four part of speech: articles, noun numbers, prepositions and verbs errors. However, Gamon *et al.* (2008)’s system has a broader scope of targeted errors than Lee’s system. Besides targeting articles, noun numbers, prepositions and verbs errors are targeted in Gamon *et al.’s* system. The system also can detect gerunds, auxiliaries, word orders, and adjectives errors. In contrast to Sun *et al.* (2007), they claim their method is capable of detecting various types of grammatical errors. My error correction system is able to provide corrections based on what corrections for ungrammatical sentences are available in a learner corpus regardless what types of error targeted.

The second issue concerns the breadth of the *coverage of a system’s domain*. Lee (2009)’s system is limited to a specific context (i.e. the flight domain), which leads to a restriction of the system grammar and vocabulary. Similarly, Gamon *et al.* (2008) implement a contextual spelling correction, so the context is also limited.

The third issue is about *suggested corrections for an erroneous sentence*. Even though various types of grammatical error can be detected in Sun *et al.’s* system, one thing that is still a matter of consideration is how to provide suggested corrections for an erroneous sentence. Sun *et al.’s* system detects whether a sentence is grammatical or not but doesn’t provide candidate corrections to an erroneous sentence. In Gamon *et al.’s* system, a list of candidate corrections is provided as well as some real-life examples extracted from the Internet. If “both error detection and suggested correction are incorrect”, they will cause particular confusion for language learners. In addition to statistical techniques, it would be useful to be able to refer to the output of a symbolic parser, that can provide independent evidence (a) that the student’s original sentence is ungrammatical, and (b) that the proposed correction is grammatical.

In language learning classrooms, corrections made by teachers are to allow language learners to become aware of errors they commit. This helps the learners to understand the errors they made, and work on improving their erroneous sentences. The language
learners have smaller grammatical knowledge and vocabularies than native speakers. In order to let the learners learn from the errors they made, error correction utterances must be easily understood by the learners. There is a lack of research focus on this issue. Lee (2009) highlights that the central problem of suggesting corrections by his system is the difficulty to determine the most appropriate word within its sentence context. One resource for an appropriate correction for a particular erroneous sentence can be provided if a list of pairs of learners’ ill-formed sentences and their suggested corrections are available. This is one of the resources I will be developing in my thesis. I will refer to this corpus as a learner corpus. In summary, a model of an error correction method, that is able to provide suggested corrections which are appropriate and easily understood by language learners is needed.

Lee (2009) performs a combination technique in error correction which relates to the error correction algorithm I will develop myself. The technique involves overgenerating and reranking sentences. In Lee’s model, the overgeneration of sentences from a reduced sentence is done arbitrarily and results in exhaustively producing many candidates of (well-formed or ill-formed) sentences. This is a weakness of the technique: the search space of possible candidate sentences is very large. The main aim of my error correction algorithm will be to narrow the search space of possible corrections by developing an explicit model of the kinds of corrections which are found. I apply language modelling techniques to statistically provide a list of suggested corrections for a particular erroneous sentence. The main resource I use to focus the search for candidate corrections is a learner corpus - in other words, a corpus of teachers’ corrections of students’ errors. Each candidate correction has a probability value that indicates how likely it is that the candidate is an appropriate correction for a certain ill-formed sentence. Then each candidate is parsed and any unparsable candidates are removed from the list. The new list is ranked by sorting a probability score assigned to each correction utterance in a descending order. The high score candidates can be represented in a form of teaching hints and provided to language learners.

Language modelling techniques are statistical methods which forecast how likely it is that a word occurs in a particular surface context. The estimation is based on the number of occurrences of the word, in the sentence, in a corpus or database. The
technique inspires me to build a model of perturbation likelihood based on a similar language modelling methodology. While a regular language model estimates the likelihood of a particular word in a given context, I develop a model of the likelihood of a particular correction in a given context. My proposed error correction technique is reminiscent of Brockett, Dolan, and Gamon’s (2006) machine-translation-inspired approach to error correction, where a probability score is calculated of one string of words being mapped onto another, using a ‘translation probability’ (\( P(S|T) \)) and a ‘language model probability’ (\( P(T) \)). But I focus just on single words in my model, rather than phrases, and I focus on the translation probability component of the statistical MT equation, generating candidate corrections whose resemblance to correct English sentences remains to be determined. My reason for focusing on words rather than arbitrarily long phrases is because some learner errors may be definable quite locally: these errors may be particularly easy to correct, and it is worth considering local techniques for identifying them, perhaps in combination with a MT-style system which maps whole phrases onto whole phrases. (They perhaps have the character of ‘low-hanging fruit’, which an error-correction system can deal with particularly accurately.)

My reason for focusing on translation probabilities rather than language model probabilities is that I envisage using my statistical correction model in conjunction with a wide-coverage symbolic parser. All candidate corrections produced by my statistical model will be parsed, and only those which are parseable will be suggested to the student. My intuition is that a symbolic parser may be more accurate at identifying well-formed English sentences than a statistical language model.

Brockett et al. use synthetic data when generating a set of phrasal translations, not data about real language learner errors. They manually generate synthetic errors with very simple templates a lot like my perturbations in reverse: e.g. converting the phrase ‘much X’ to ‘many X’. My perturbation data are generated automatically from a more naturalistic learner corpus that I gathered from real ESL learners (described in Chapter 3). The creation of the perturbation data is described in §4.4.

My statistical model of error correction is able to propose candidate corrections to ill-formed sentences, similarly to Gamon, Gao, Brockett, and Klementiev’s and Lee’s model. While my model implements Katz backoff (Katz, 1987) technique with Witten-
Bell discounting (Witten and Bell, 1991), Gamon et al. (2008) implement backoff technique with Kneser-Ney discounting (Kneser and Ney, 1995). The difference between Witten-Bell and Kneser-Ney is the former focusses on how many words are seen for the first time and the latter pays attention to the number of contexts in which a word occurs. However, my error correction model is a scoring model not a probabilistic model. More detailed on my error correction model will be given in §4.3.1. The main problem that Gamon et al. (2008) seem to have is when the candidate corrections are also incorrect. These are known as false positives. In my error correction system, candidate corrections that are generated from the language model are parsed using a wide-coverage symbolic grammar. Therefore only candidates that are correctly parsed will be suggested.

The error correction techniques in Lee (2009)’s system involve sentence overgeneration and reranking processes. The latter ranks and scores candidate corrections by using stochastic grammar models. The former process overgenerates candidate corrections from the initial incorrect sentence. The process is done arbitrarily and results in exhaustively producing many candidates of (well-formed or ill-formed) sentences. This is a weakness of the technique: the search space of possible candidate sentences is very large. An explicit model of the kinds of corrections which are found by my model, narrows the search space of possible corrections. This is the main novelty of my model: it explicitly represents ‘corrections’ — i.e. circumstances where a language teacher corrects a student’s language. Most statistical error correction models are simply models of the target language being taught; their aim is just to define the kinds of sentence which are expected in the target language. These models are good at recognising when a student’s utterance contains an error. But they are not so good at providing suggestions about how to correct errors. In any student’s sentence, there are many things which could be changed: the space of possible corrections is too large to be exhaustively searched. Therefore, a statistical system (such as my error correction model) which explicitly models the incidence of corrections can help guide the search for good corrections.
Table 2.10: Types (dialogue-based or parser-based) of CALL systems

<table>
<thead>
<tr>
<th></th>
<th>Dialogue-based</th>
<th>Parser-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kaitito</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>L2Tutor</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>SPELL</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Let’s Chat</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>German Tutor</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>ICICLE</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>Arabic ICALL</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>BANZAI</td>
<td></td>
<td>√</td>
</tr>
</tbody>
</table>

2.8 Summary

This chapter reviewed current research topics related to CALL systems. First, 3 phases of CALL were discussed, followed by theories of SLA that are linked to the CALL phases. Then, I introduced two types of CALL systems: dialogue-based CALL and parser-based CALL systems, by discussing some existing systems. Table 2.10 summarises each system either as dialogue-based type, parser-based type, or both types. I then explained two other relevant topics which are types of errors that language learners make, and corrective feedback. Table 2.11 outlines the types of CF provided by the CALL systems I have described earlier in §2.3 and §2.4.

Since my main focus is on automatic error correction in dialogue-based CALL systems, two approaches to error correction were discussed: grammar-based methods and statistical methods. Based on the discussion in §2.7.3, I am interested in implementing a combination of error correction techniques in a dialogue-based CALL system. The techniques are a statistical surface-based error correction module and a wide coverage parsing system. While the parser is capable of accepting grammatical sentences and rejecting ungrammatical sentences, the surface-based error correction technique is capable of providing an error diagnosis for an ill-formed sentence. The error diagnosis

---

19 I utilise the Kaitito dialogue-system CALL when implementing an error correction system.
Table 2.11: Types of corrective feedback provided by CALL systems

<table>
<thead>
<tr>
<th></th>
<th>Kaitito</th>
<th>L2Tutor</th>
<th>SPELL</th>
<th>Let’s Chat</th>
<th>German Tutor</th>
<th>ICICLE</th>
<th>Arabic ICALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metalinguistic</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Recast</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Explicit correction</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Acknowledgement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>No corrective feedback</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

consists of a list of candidate corrections for the erroneous sentence, where the list is statistically generated from an existing learner corpus. The learner corpus consists of a collection of pairs of an ill-formed sentence and its suggested correction.

In the following chapter (Chapter 3), an empirical study is carried out to investigate what common errors are committed by the learners and how to tackle the errors. The output of the study is a learner corpus that becomes a valuable resource for the statistical error correction system that I will propose later in Chapter 4.
Chapter 3

An empirical study of learners’ errors in written language learning dialogues

3.1 Introduction

This chapter describes the first stage of my research progress: a study of learners’ error responses to some typical questions or dialogues in daily conversation. The data gathered here forms the corpus used by the statistical model of error correction that I propose in Chapter 4. In this chapter I describe how I gathered the data, and also give a preliminary analysis of the data, to gain an idea of what are the most common errors that learners committed. This analysis should be useful to anyone embarking on developing a CALL system for Malaysian EFL students. Of course, the data should also be of interest to EFL teachers in Malaysia, providing information about the patterns of their students’ errors and how these change over time.

Before I present my own study, I will describe two existing studies of errors in Malaysian EFL learners. There are not many such studies, but I will mention two: that of Jalaluddin, Awal, and Bakar (2008) and that of Maros, Hua, and Salehuddin (2007). The purpose of the first study is to investigate how environmental and linguistic issues affect EFL learning. Participants of this study are secondary school pupils in
Malaysia: English is their second or foreign language. Jalaluddin et al.’s study analyses two cloze tests which are given to 315 Form 2 students. The tests cover morphology and syntax; the format of questions is multiple choice. Here I only report results of the study for the linguistic factor. In the morphology structure, some of the areas tested are affixes, adverbs, plural form, and superlative. Importantly, Jalaluddin et al. (2008) do not present an explicit taxonomy of error types within which they frame their results. However, they do give some indication of the most common error types. Among the four areas, the most common error that the students made is in use of the plural form which contributes about 74%, followed by superlative (72%), use of the affixes (64%), and adverbs (56%). These percentage values are the percentage number of students who committed to the errors. In the syntax form, areas which are covered are subject-verb agreement, the copula ‘be’, articles, determiners, and relative pronouns. The highest number of errors committed by the students is relative pronouns (82%), followed by subject-verb agreement (76%), the copula ‘be’ (67%) and articles (64%). Jalaluddin et al. (2008) suggest that these errors are due mainly to L1 transfer problems.

Maros et al. (2007) perform their study to investigate how interference (or transfer problem) affects EFL learning in Malaysian students. This study analyses errors in two essays written by Form 1 students in six schools. The students are given choices to pick two topics from a list of topics provided. The two most selected topics are “My best friend” and “My family”. Results show that determiners, subject-verb agreement and the copula ‘be’ are the three most problematic grammatical errors committed by the participants. Maros et al. claim these errors are due to students’ L1 transfer.

These two existing studies have similarities with my empirical study. The first similarity is the nationality of participating students who are all Malaysians. The second one is my study and the two studies have investigated the most common errors made by the students. Nevertheless, there are some differences between my study and the other two. Firstly, I develop an explicit taxonomy of error categories for Malaysian EFL students. Secondly, both existing studies only focus on one school level, or form, while my study covers various form levels, so I can investigate the performance of students over time. Thirdly, I provide quantitative results, using statistical tests to
check for significant trends.

In §3.2, I will explain the format of the study, discussing the participants involved, and materials and methods used. Existing error classification schemes will be described in §3.3 followed by my scheme of error classification in §3.4. This scheme is referred to during the error annotation process. In §3.5 I will outline the procedures used for error annotation tasks using my error classification scheme. To ensure data reliability and validity of the scheme, inter-coder agreement tests are applied, as described in §3.6. §3.7 will show results of the agreement tests. Lastly in §3.8, I will describe the results of the study after the annotation tasks are completed.

3.2 Format of the Study

3.2.1 Subjects

Subjects of the study were pupils in secondary schools in Malaysia. Three secondary schools were involved; in each school, pupils from Form 1, Form 2 and Form 3 classes participated in the study. These three schools are categorised as suburban schools. The students’ ages were between 13 and 15 years old. These students have had six to eight years of learning English. In Malaysian secondary schools, each form level may have more than one class which is known as a “stream”. Pupils are allotted to a certain stream based on their Ujian Penilaian Sekolah Rendah (UPSR)\(^1\) examination results. Some schools named the stream as “RK”, “A”, “B”, “C” which means students of stream RK has better results than stream A, and students in stream A are better than stream B and students in stream B are better than stream C’s students. The total of students was 173. Table 3.1 details numbers of students, by school, form and stream. See Appendix B for further information about teaching and learning English in Malaysia.

Malaysia has three main races: Malay, Chinese and Indian. The distribution of

---

\(^1\)This is an examination which is compulsory for Standard 6 pupils before they can proceed to secondary school. The examination comprises of core subjects such as Malay Language, English, Mathematics and Science.
Table 3.1: Numbers of students, by school, form and stream

<table>
<thead>
<tr>
<th>School</th>
<th>Form 1</th>
<th>Form 2</th>
<th>Form 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Num</td>
<td>Stream</td>
<td>Num</td>
</tr>
<tr>
<td>Sch1</td>
<td>34</td>
<td>1RK</td>
<td>30</td>
</tr>
<tr>
<td>Sch2</td>
<td>5</td>
<td>1A</td>
<td>14</td>
</tr>
<tr>
<td>Sch3</td>
<td>4</td>
<td>mix</td>
<td>44</td>
</tr>
</tbody>
</table>

Table 3.2: A composition and percentage across races

<table>
<thead>
<tr>
<th></th>
<th>Malay</th>
<th>Chinese</th>
<th>Indian</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sch1</td>
<td>87.64%</td>
<td>3.37%</td>
<td>8.89%</td>
</tr>
<tr>
<td>Sch2</td>
<td>42.86%</td>
<td>28.57%</td>
<td>28.57%</td>
</tr>
<tr>
<td>Sch3</td>
<td>30.36%</td>
<td>66.07%</td>
<td>3.57%</td>
</tr>
<tr>
<td>Average</td>
<td>61.85%</td>
<td>27.75%</td>
<td>10.40%</td>
</tr>
</tbody>
</table>

race in my sample is outlined in Table 3.2. Although the Malay language is the national language of Malaysia, every race has its own mother-tongue or L1. Malay people speak the Malay language, Chinese speak Mandarin, and Indian people speak Tamil. However, Malay is the language medium in teaching all subjects except English, Mathematics and Science. In this study, the majority of the students are Malay, making up almost 62% of the total sample. However this percentage is not high enough to be able to draw strong conclusions about the likely origins of any errors in transfer issues from Malay. In §3.8.1, when I consider the issue of transfer errors, I will look at Sch1 separately, where there is a large majority of Malay students. But for other analyses I am interested in the incidence of error types in the general population, and I will group the three schools together, to give a good reflection of all three ethnicities. In §3.8.2 when considering longitudinal effects I will again focus on Sch1.
### Table 3.3: The Types of Pronoun

<table>
<thead>
<tr>
<th>Pronoun Types</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>First singular</td>
<td><em>I</em></td>
</tr>
<tr>
<td>Third singular</td>
<td><em>he, she</em></td>
</tr>
<tr>
<td>First plural</td>
<td><em>we</em></td>
</tr>
<tr>
<td>Third plural</td>
<td><em>they</em></td>
</tr>
</tbody>
</table>

#### 3.2.2 Materials and Methods

The study required students to write answers to a list of English questions. The list consisted of 45 questions. It included both Wh-form\(^2\) and open-ended questions. In the questions, three types of grammatical structure were targeted:

1. **Tenses** (present, past and future),

2. **Pronouns** (1st singular (*I*), 3rd singular (*he, she*), 1st plural (*we*), and 3rd plural (*they*)), and

3. **Subject-verb agreement**.

Pronouns and subject-verb agreement are obviously related and were targeted together. Jalaluddin *et al.* (2008); Maros *et al.* (2007) report that subject-verb agreement is one of the most problematic grammatical errors committed by students. The questions asked common things about a student, her/his parents, and her/his best friends. The list of questions together with students’ responses is valuable because it becomes a resource for me in evaluating a model of error correction I proposed in Chapter 4. Some of the questions will be used again as questions posed to students in a dialogue-based CALL that I developed as discussed in Chapter 5. The list of questions is attached in Appendix C.

The questions were submitted to English teachers. They distributed the questions on paper to their students. The students had to complete the questions during class.

---

\(^2\)Wh-form questions are questions which their first word starts with *what, when, where, which, who, why, whose, whom,* or *how.*
The students were asked to answer all questions. The teachers collected back the answered questions and returned them to me.

The next task was to transcribe all responses into a spreadsheet. Table 3.4 shows some examples of the transcribed data. The data are represented in a tabular form in which from left to right each column represents the students’ number, their schools’ unique identification (id), their class form, the grammatical type of the question, the question number, the question itself and the students’ response to the question, respectively. For this data, 173 students responded to 45 questions. A total of 7785 sentences (173 students × 45 questions) were collected.

In order to check the grammaticality of each response, a list of error categories is required. The list becomes a reference to mark or annotate any errors that occurred in

<table>
<thead>
<tr>
<th>Student Id</th>
<th>School Id</th>
<th>Form Level</th>
<th>Grammar Type</th>
<th>Question Number</th>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3</td>
<td>Sch1</td>
<td>1</td>
<td>Present Tense</td>
<td>3</td>
<td>Tell me about your country.</td>
<td>My country is beautiful.</td>
</tr>
<tr>
<td>S12</td>
<td>Sch1</td>
<td>1</td>
<td>Past Tense</td>
<td>9</td>
<td>Where did you go on your last school holiday?</td>
<td>Go to camping.</td>
</tr>
<tr>
<td>S20</td>
<td>Sch1</td>
<td>1</td>
<td>Future Tense</td>
<td>3</td>
<td>What will your parents do this evening?</td>
<td>They will cooking.</td>
</tr>
</tbody>
</table>
the responses. Such a list is also known as an error classification or coding scheme. I will first explain some existing coding schemes, and then describe the error classification scheme which I created myself in the next section (§3.4).

3.3 Existing Error Classification Schemes

The data that I collected will be referred to as the learner corpus. The learner corpus consists of grammatical and ungrammatical sentences written by language learners. Error analysis can be performed quantitatively by investigating the learners’ errors on the corpus data. Jie (2008) listed three important aspects of error analysis in SLA. Firstly, the learners’ errors tell the teachers how far towards the goal the learners have progressed and what remains for them to learn. Secondly, the learners’ errors provide evidence of how language is learned or acquired, for researchers. Lastly, the learners’ errors are means whereby learners test hypotheses about the interlanguage of the learners. In my study, firstly, I analysed my corpus data to identify most frequent errors committed by students and secondly to investigate the performance of the students across form levels. This relates to the first aspect indicated by Jie. As a first step in the analysis, I went through each sentence in the corpus and corrected any syntax errors, if any. In order to do this, a scheme of error classification was required to be used as a reference to annotate all the located errors.

The creation of error classification schemes relies on error taxonomies which contain categories for error classification. As agreed by James (1998), there are two dimensions which should be included in error taxonomies. The dimensions are

- a linguistic category classification, which represents linguistic features of learner error for example tense, grammar, lexical, etc., and

- a target modification taxonomy, which accounts for what actions need to be done to correct learners’ errors for instance insertion, deletion, replacement, order, etc.

In the next subsection, I will explain three existing error classification schemes used to annotate learners’ errors in corpora. The schemes are the Cambridge Learner Corpus, the National Institute of Information and Communications Technology Japanese
Learner of English, and the FreeText project. I will use the terms “error classification scheme”, “error coding scheme” interchangeably; and like also the terms “error codes” and “error tags”. In the last subsection, I will discuss a spelling error technique proposed by Kukich (1992) which I adopted in my error classification scheme.

3.3.1 The Cambridge Learner Corpus

The Cambridge Learner Corpus (CLC) is a collection of written essay examination scripts taken by learners where English is their second or foreign language (Nicholls, 2003). The scripts are transcribed and ranged across 8 EFL examinations which cover both general and business English. According to Nicholls (2003), CLC is growing but at the time reported, it consists of 16 million-word. Only 6 million-word component of the corpus has been error coded. The coding of errors is based on an error classification scheme developed at Cambridge University Press.

The error classification scheme covers 80 types of error and has 31 error codes. Each error code is represented in an eXtension Markup Language (XML) convention as shown below:

```xml
<#CODE> wrong word corrected word < /#CODE >
```

In most of the error codes, `<#CODE>` examples are based on a two-alphabet system: the first letter represents the general type of error, and the second one identifies the word category of the required word. The general types of error consist of a wrong form used (F), a word missing (M), replacement of a word (R), an unnecessary word (U) and an incorrectly derived word (D). Beside the general types, other types of error included are countability (C), false friend (FF), and agreement (AG) errors. Some additional errors such as spelling error (S), American spelling (AS), wrong Tense of Verb (TV), incorrect verb inflection (IV) are also included.

There are 9 word categories such as pronoun (A), conjunction (C), determiner (D), adjective (A), noun (N), quantifier (Q), preposition (P), verbs (V), and adverb (Y). Punctuation errors are also included and represented as P in the second letter of the error code following the general types of errors M, R, U as the first letter. Below is
an example of a sentence with a correction using the CLC error classification scheme (Nicholls, 2003, pp576):

So later in the evening I felt <#RY>hardly|seriously</#RY> ill.

The above error code annotation means “Replace (R) the adverb (Y) word “hardly” with a more appropriate adverb, “seriously””. The CLC only has one punctuation category which caters for all types of punctuation. The two-alphabet error code system is in flat representation which means CLC does not allow for identification of errors at different levels of specificity. Flat annotation is unsuitable for the inclusion of additional interpretation of errors since once annotations are added alongside the errors, additional interpretation layers of annotation cannot be inserted (Díaz-Negrillo and Fernández-Domínguez, 2006).

3.3.2 The National Institute of Information and Communications Technology Japanese Learner of English

The National Institute of Information and Communications Technology Japanese Learner English (NICT JLE) corpus is a two-million-word speech corpus from Japanese who are learning English (Izumi, Uchimoto, and Isahara, 2005). Its source is from 1281 audio-recorded speech samples of an English oral proficiency interview test ACTFL-ALC Standard Speaking Test (SST). The NICT JLE error classification scheme has 46 error tags which have three pieces of information: POS, morphological/grammatical/lexical (MGL) rules, and a corrected form (Izumi et al., 2005, pp75). Similar to CLC, the error code of NICT JLE is also represented in a XML form. Below is an example of error codes:

\[<P_G\quad crr = "corrected\quad word"\quad >\quad wrong\quad word\quad <\quad /P_G>\]

The P symbol identifies a POS symbol (i.e. n for noun) and G symbol represents the MGL rules (i.e. num for number which is under the grammatical system). There are 11 categories of POS in the NICT JLE such as noun, verb, modal verb, adjective, adverb, preposition, article, pronoun, conjunction, relative pronoun, and interrogative. In addition, there is one more error category which is named with Others. This category
represents errors such as Japanese English, collocation, misordering of words, unknown type errors, and unintelligible utterance.

Below is an example of a sentence with a correction using the NICT JLE error classification scheme (Izumi et al., 2005, pp75):

\[ I \text{ belong to two baseball } \text{teams} \text{ team}. \]

The NICT JLE doesn’t cater for any punctuation errors. As suggested by James (1998), in a creation of error classification schemes, two dimensions of error taxonomy must be included. However, NICT JLE includes one only which is a linguistic category classification. The excluded dimension is target modification taxonomy, but with an exception to one error code which is Misordering of words. The NICT JLE is also considered as L2-biased because it has a relative pronoun tagset which only occurs in the Japanese language. Similar to the CLC, the NICT JLE error codes representation is flat and it does not allow for identification of errors at different detailed levels.

### 3.3.3 The FreeText System

The FreeText system is an error annotation system used to annotate the French Inter-language Database (FRIDA) corpus. The FRIDA corpus contains a large collection of intermediate to advanced L2 French writing. It contains 450,000 words, but only two-thirds have been error annotated completely (at the time Granger (2003) reports).

FreeText consists of three levels of annotation: error domain, error category, followed by word category. The error domain specifies whether the error is formal, grammatical, lexical, and so forth. There are nine error domains such as form (<F>), morphology (<M>), grammar (<G>), lexis (<L>), syntax (<X>), register (<R>), style (<Y>), punctuation (<Q>), and typo (<Z>). Each error domain has its own error categories. For example for the morphological error domain, there are 6 error categories. As for the second level, the number of error categories from each error domain ranges between 2 to 10 categories with a total of 36 categories. An exception is the (<Z>) error domain because no error categories are included.

The word category consists of a POS type which comprises 11 major categories: for example adjective, adverb, article, conjunction, determiner, noun, preposition, pro-
noun, verb, punctuation, and sequence. The number of POS sub-categories from each major category ranges between 1 to 12 categories with a total of 54 subcategories.

For each error, an annotator has to select 3 tags from the three different groups (the error domain, the error category and the word category). There are 9 tags from the error domain, 36 tags from the error categories and 55 POS tags. Therefore, in total there are about 100 error tags. Below is an example of a sentence with a correction using FreeText (Granger, 2003, pp470):

L’héritage du passé est très <G><GEN><ADJ>#forte </ADJ></GEN></G> et le sexisme est toujours présent.

3.3.4 Spelling Correction Techniques

In this section, I will explain a technique in automatic spelling correction which I adopted in the creation of my error classification scheme. Kukich (1992) in her paper, discusses the current state of various techniques for correcting spelling errors in three areas of research: nonword error detection, isolated-word error correction and context-dependent word correction.

In response to the nonword error detection area, efficient pattern matching and n-gram analysis techniques have been developed for detecting strings that do not appear in a given word list. The context-dependent word correction uses NLP tools. The isolated-word correction focuses on detailed studies of spelling error patterns. Kukich identifies four common error types of isolated-word correction: insertion of a character, deletion of a character, substitution of a character with another character, and transposition of two adjacent characters. The four error correction techniques are similar to the target modification taxonomy which will be mentioned in the next section regarding the creation of my error classification scheme (§3.4). Therefore I decided to adopt those error correction types as one of error codes in my error classification scheme. See Table 3.5 for the comparison between a word level and a syntax level error correction.
Table 3.5: Common error types identified by Kukich (1992)

<table>
<thead>
<tr>
<th></th>
<th>Word Level</th>
<th>Syntax Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert correction</td>
<td>speling → spelling</td>
<td>I from Malaysia. → I am from Malaysia</td>
</tr>
<tr>
<td>Delete correction</td>
<td>scarry → scary</td>
<td>My parents they are kind. → My parents are kind.</td>
</tr>
<tr>
<td>Substitution correction</td>
<td>sorri → sorry</td>
<td>My city is peace. → My city is peaceful.</td>
</tr>
<tr>
<td>Transposition correction</td>
<td>taht → that</td>
<td>I like that car blue. → I like that blue car.</td>
</tr>
</tbody>
</table>

3.4 My Error Classification Scheme

Each error classification scheme has its own benefits and weaknesses and has been developed according to the goals and objectives of the research. Certain variables such as the learners’ target language, the learners’ L1 background or size of the corpus differ across learners’ corpora, and therefore they may have an effect on error codes being built (Díaz-Negrillo and Fernández-Domínguez, 2006). The mother tongue of students is obviously very important too. My error classification scheme is developed to suit my purpose of study. Differences I want to highlight here are in terms of the learners’ mother tongue, corpus data resources, scope of error types, structure of error codes, and how error codes are annotated.

Firstly, my error classification scheme was developed for the purpose of annotating errors committed by Malaysian learners of English. The existing error classification schemes described earlier were for Japanese, French and Dutch EFL learners. NICT JLE is the corpus for Japanese learners of English. In the FreeText system, there are three different categories of learners who are learning French: English, Dutch, and learners who have different L1. The CLC is the English corpus of learners who are from 86 different mother tongues. While in retrospect I could probably have used the CLC error taxonomy, I wanted to develop a taxonomy customised to my target group of language learners, as several other researchers have done for different languages.
A second difference is that my error classification scheme is developed to analyse data gathered in a particular format. The learners’ data is a collection of written answers, according to written posed questions targeting particular grammatical constructions. The CLC data comprises English examination written essay scripts. The FRIDA corpus consists of a collection of data of L2 French writing. In contrast, NICT JLE is a speech corpus of Japanese who are learning English the source of which is from audio-recorded speech samples of an English oral proficiency interview test ACTFL-ALC SST.

A third difference is related to types of error covered in the corpus. Almost all error codes in the existing schemes are included in my scheme. Additionally, I provided other codes to cater for dialogue errors. The error codes are; for any students’ responses which are

1. considered as irrelevant answers to its given questions, *unrelated answers*,

2. considered as *incomplete answers*,

3. provided partly, *partial answers* and

4. not supplied by the students, *no response*.

My error classification scheme and the existing schemes except the NICT JLE corpus cater for punctuation errors. According to Díaz-Negrillo and Fernández-Domínguez (2006), NICT JLE and CLC are biased to L2 or other English accent. For example, NICT JLE has an error code for English relative pronouns because the Japanese language does not have relative pronouns. The CLC system has an error code to represent an usage of American Spelling words.

A final difference concerns the codes I use to report errors. While the error codes of existing schemes are in XML form, and are annotated on specific parts of sentences, my annotations use a predicate-argument structure, and are coded as attributes of whole sentences. For instance a sentence may be coded as ‘delete(*X*)’ where delete is the predicate and X is the argument. The combinatorial aspect of my predicate-argument scheme is something which can be reproduced in XML - for instance the FreeText scheme allows error categories to be parameterised in a similar way. XML
annotations have the additional benefit of being localised to particular places within sentences, so these annotations are more detailed than those which I use. However, in the automated error correction scheme I describe in Chapter 4, errors can be precisely localised to particular places in sentences. The goal of the error annotation study in the current chapter is mainly to get a feeling for which are the most common errors made by the target student population, rather than to develop an automated mechanism for correcting these errors.

As mentioned earlier, each error tag in my scheme is represented in a form of predicates. Each predicate has two arguments at most. The error scheme has two levels: a coarse level and a fine level. The coarse level is represented by the predicate of error tags. This represents the rough idea of what types of error are involved. In the fine level, arguments of predicates are determined. Here, what types of linguistic forms or POS involved in the error are obtained. In contrast, the structure of error codes’ annotation in the CLC and NICT JLE corpora is flat, i.e. involves only one level of annotation. Nevertheless, the FreeText system classify errors at three different levels of specificity which facilitates application of their most general categories, and adaptation of the error tags according to the language needs.

In all existing schemes, their error tags are coded alongside the errors. On the other hand, the error codes of my scheme aren’t tagged within ungrammatical sentences. Each error code is assigned to its respective allocated column. The columns are located between an ungrammatical sentence and its respective correct sentence. In addition, at most 4 error codes can be assigned. If there are more error codes required, the respective sentences are tagged as “unclassifiable errors”.

In my scheme, I also adopted a spelling correction technique proposed by Kukich (1992). While Kukich uses insert, delete, substitution and transposition methods for correcting spelling errors I applied the technique to correct syntax errors of sentences. Development of my error classification scheme began in parallel with the transcribing process. While transcribing the data, I listed out any errors that occurred. Based on the list, I came out with a first draft of the error categories scheme. Then I tested the scheme with the data. When there were certain errors which could not be fitted into any existing categories, another category was added. This process was repeated until
no more categories were added. In the following section, I will explain the features of an effective error coding scheme. Then in §3.4.2, a detailed description of the error tags used in my error classification scheme is outlined.

3.4.1 Effectiveness of Error Classification Schemes

Before explaining the list of error tags available in my error classification scheme, I would like to discuss the features an error classification scheme should have. Granger (2003) proposes four features any error classification scheme must possess to be fully effective. The features are consistent, flexible, reusable, and informative which are also mentioned in Díaz-Negrillo and Fernández-Domínguez (2006).

An error classification scheme must be consistent in such a way that persons who are using it (later, we refer them to annotators), are able to provide similar judgements about errors. This means the annotators should have a high agreement level in their judgement. As such, the scheme should have detailed descriptions of error categories and error tagging principles. To assess the consistency of the scheme, it is important to use inter-annotator reliability tests such as Kappa and Alpha. I applied these reliability tests. Validity was established in this way (explained in §3.6).

Flexibility means error tags are able to be deleted or inserted during annotating. The error tags should be easy to retrieve even after the annotation stage. The flexibility of error tags retrieval indicates how many levels of error code specificity can be obtained. My error codes are represented in two hierarchical levels: predicate and argument. The predicate identifies the error types, and the argument determines the linguistic categories involved.

The scheme should also be informative but manageable. Informative means the scheme should provide detailed descriptions about error categories. But, too detailed may become unmanageable for annotators. My scheme, which involves both fine-grained and coarsed-grained error annotations, fulfils this requirement: coarse-grained annotations are provided by predicates, and fine-grained detail is provided by their arguments.
3.4.2 Error Tag Categories

There are six categories of error tags in my error classification scheme as shown below:

1. Agreement errors
2. Tense errors
3. Spelling and Vocabulary errors
4. Delete/Insert/Substitute/Transpose
5. Dialogue errors
6. Unclassifiable error

As mentioned earlier, each error tag is represented in a form of predicate. The predicate form has at from 0 (zero) to 2 arguments. The full list of error categories is attached in Appendix D. The following section will explain each error category.

3.4.2.1 Agreement Errors

The agreement errors are divided into 3 sub-categories as below:

- subject-verb agreement errors: \( sva(X) \),

- determiner noun agreement errors: \( \text{det-n-ag} \), and

- noun number errors: \( \text{noun-num-err} \).

Subject verb agreement errors are represented by a \( sva(X) \) tag and can be examined according to which type of verb appears in a sentence. An argument \( X \) is the verb which ranges over an open class verb, the copula be and have. A determiner noun agreement error happens when there is no agreement between a determiner and a noun. This category has no arguments. The focus is always on the noun. Noun number errors are specific to generic nouns, when they are given as bare singular nouns rather than bare plural – a common error for Malaysian EFL learners. Again, the category has no arguments.

Table 3.6 depicts the detailed information of each sub-category of the agreement errors. Some examples of the error tags based on the raw data are shown in Table 3.7.
Table 3.6: The Agreement Error Tags

<table>
<thead>
<tr>
<th>Category</th>
<th>Predicate symbol</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject-verb agreement</td>
<td>sva((X))</td>
<td>(X) can be either (be), (have) or (verb) (meaning an open class verb)</td>
</tr>
<tr>
<td>Determiner noun agreement</td>
<td>det-n-ag</td>
<td>none</td>
</tr>
<tr>
<td>Noun number errors</td>
<td>noun-num-err</td>
<td>none</td>
</tr>
</tbody>
</table>

3.4.2.2 Tense Errors

The third category is tense errors which are represented as a \texttt{tense-err(}\(X, Y)\texttt{)} tag. The error tag means an incorrect tense is used and an argument \(X\) must be replaced by an argument \(Y\). The arguments range over the following tenses:

- present tense \((\texttt{pres})\),
- infinitive tense \((\texttt{inf})\),
- past tense \((\texttt{past})\),
- present progressive tense \((\texttt{progr})\), and
- past participle tense \((\texttt{past-p})\).

Some examples of the tense error tags based on the raw data are shown in Table 3.8.

3.4.2.3 Spelling Errors

The spelling error is represented as \texttt{spell-err} tag. Table 3.9 shows some examples of different types of error which are analysed as involving mis-spelled words.

3.4.2.4 Vocabulary Errors

The vocabulary error is represented as a \texttt{vocab-err} tag. All non-English words such as Malay words including mis-spelled ones are considered vocabulary errors. Some ex-
Table 3.7: Error annotation using *sva*, *det-n-ag*, and *noun-num-err* tags

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
<th>Corrected Response</th>
<th>Error Tag Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where were you born?</td>
<td>I were born in Hospital Kem Terendak.</td>
<td>I <em>was</em> born in Hospital Kem Terendak.</td>
<td><em>sva</em>(be)</td>
</tr>
<tr>
<td>What is your best friend’s ambition?</td>
<td>She want to be a doctor.</td>
<td>She <em>wants</em> to be a doctor.</td>
<td><em>sva</em>(verb)</td>
</tr>
<tr>
<td>What is your father’s job?</td>
<td>My father is an engineer.</td>
<td>My father is an engineer.</td>
<td><em>det-n-ag</em></td>
</tr>
<tr>
<td>What does your best friend like together?</td>
<td>He likes to read book.</td>
<td>He likes to read <em>books</em>.</td>
<td><em>noun-num-err</em></td>
</tr>
</tbody>
</table>

Table 3.8: Error annotation using *tense-err*(X,Y) tags

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
<th>Corrected Response</th>
<th>Error Tag Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>What did you do last weekend?</td>
<td>I go to Kuala Lumpur last weekend.</td>
<td>I <em>went</em> to Kuala Lumpur last weekend.</td>
<td><em>tense-err</em>(pres,past)</td>
</tr>
<tr>
<td>What do your parents like doing?</td>
<td>They like to gardening.</td>
<td>They like <em>to garden</em>.</td>
<td><em>tense-err</em>(progr,inf)</td>
</tr>
</tbody>
</table>
Table 3.9: Wrongly spelled word types

<table>
<thead>
<tr>
<th>Type of words</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>incorrect word spelling</td>
<td>peacefull, intresting, picknic, realitif, televisyion, bycycle, libary, scholded, borrar, telivisian, plis, teachear</td>
</tr>
<tr>
<td>mis-spelled verb words in past tense</td>
<td>eated, borned, maked, teached</td>
</tr>
<tr>
<td>incorrect plural words spelling</td>
<td>radioes</td>
</tr>
<tr>
<td>a word that should be separated</td>
<td>eventhough, bestfriend</td>
</tr>
</tbody>
</table>

Some examples of Malay words are merempit, pasar minggu, tuisyen, and juruteknik. However, an exception is made for non-English proper names. These are not errors at all.

3.4.2.5 Delete/Insert/Substitute/Transpose Tags

If an error cannot be classed in one of the above categories, I revert to a simpler, surface-based error classification scheme, which just describes the manipulations needed in order to fix the sentence, referring to the parts of speech of the words involved.

The delete tag is represented as del(\(X\)) which means we need to delete \(X\) to fix the sentence. The insert tag is represented as ins(\(X\)) which means we need to insert \(X\). The substitute tag is represented as subst-with(\(X, Y\)) which means we need to substitute \(X\) with \(Y\). The transpose tag is represented as transp(\(X, Y\)) which means I need to transpose \(X\) and \(Y\). The arguments \(X\) and \(Y\) can be any one of the linguistic forms or part of speech (POS) depicted in Table 3.10.

Some examples of error annotations using the tag, based on the raw data, are shown in Table 3.11. As shown in this table, a column “Corrected Response” represents a corrected version of an original response³.

³Emphasis has been placed on the inserted, substituted and transposed words in these examples only; the actual corpus is performed in tabular form and there is no emphasis on the corrected versions.
<table>
<thead>
<tr>
<th>Linguistic Forms</th>
<th>Symbols Used</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>noun</td>
<td>noun</td>
<td><em>school, name, city</em></td>
</tr>
<tr>
<td>verb</td>
<td>verb</td>
<td><em>go, like, celebrate</em></td>
</tr>
<tr>
<td>adjective</td>
<td>adj</td>
<td><em>beautiful, good, peaceful</em></td>
</tr>
<tr>
<td>adverb</td>
<td>adv</td>
<td><em>slowly, quickly</em></td>
</tr>
<tr>
<td>the copula <em>be</em></td>
<td>be</td>
<td><em>is, are</em></td>
</tr>
<tr>
<td>pronoun</td>
<td>pron</td>
<td><em>I, he, they, we</em></td>
</tr>
<tr>
<td>proper noun</td>
<td>prop-n</td>
<td><em>Nora, Johan</em></td>
</tr>
<tr>
<td>have</td>
<td>have</td>
<td><em>have, has</em></td>
</tr>
<tr>
<td>definite determiner</td>
<td>def-det</td>
<td><em>the</em></td>
</tr>
<tr>
<td>indefinite determiner</td>
<td>indef-det</td>
<td><em>a, an</em></td>
</tr>
<tr>
<td>possessive determiner</td>
<td>poss-det</td>
<td><em>my, her, his, their</em></td>
</tr>
<tr>
<td>possessive morphology</td>
<td>poss-morph</td>
<td><em>'s, s’</em></td>
</tr>
<tr>
<td>conjunction</td>
<td>conj</td>
<td><em>and, or</em></td>
</tr>
<tr>
<td>modal auxiliary</td>
<td>modal-aux</td>
<td><em>can, must</em></td>
</tr>
<tr>
<td>infinitive marker to</td>
<td>inf-mrkr-to</td>
<td>I want to... He likes to...</td>
</tr>
<tr>
<td>apostrophe</td>
<td>apstrpe</td>
<td><em>’</em></td>
</tr>
<tr>
<td>clause delimiter</td>
<td>clse-deltr</td>
<td>full stop (.), hyphen (-),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>question mark (?), and comma (,)</td>
</tr>
<tr>
<td>will</td>
<td>will</td>
<td>I will go</td>
</tr>
<tr>
<td>verb phrase,</td>
<td>VP</td>
<td><em>wake up</em></td>
</tr>
<tr>
<td>noun phrase</td>
<td>NP</td>
<td><em>my name</em></td>
</tr>
<tr>
<td>morphology</td>
<td>morph</td>
<td>same root word for example young to young(<em>er</em>)</td>
</tr>
<tr>
<td>preposition</td>
<td>from, at, in, for, to, of, with, on</td>
<td>some common preposition words</td>
</tr>
</tbody>
</table>
Table 3.11: Error annotation using delete/insert/substitute/transpose tags

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
<th>Corrected Response</th>
<th>Error Tag Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is your best friend’s name?</td>
<td>My best friend it is Khairul.</td>
<td>My best friend is Khairul.</td>
<td>del(pron)</td>
</tr>
<tr>
<td>What is your name?</td>
<td>My name Nora</td>
<td>My name is Nora</td>
<td>ins(be)</td>
</tr>
<tr>
<td>What do you want to be when you grow up?</td>
<td>I want to be a greatest scientist.</td>
<td>I want to be a great scientist.</td>
<td>del(morph)</td>
</tr>
<tr>
<td>Tell me about your city.</td>
<td>My city it very good.</td>
<td>My city is very good.</td>
<td>subst-with(pron,be)</td>
</tr>
<tr>
<td>What do you and your friends do together?</td>
<td>We make a group study.</td>
<td>We make a study group.</td>
<td>transp(noun,verb)</td>
</tr>
</tbody>
</table>
### 3.4.2.6 Dialogue Errors

The sixth category is dialogue errors which are divided into four types:

- unrelated-answer: *unrel-ans*,
- incomplete-answer: *incomp-ans*,
- partial-answer: *part-ans*, and
- no response: *no-resp*.

A **unrelated answer** tag is given when there occurs a question comprehension error, misunderstanding or giving a different answer to a given question. When a student did not answer the question wholly, the response is marked as **part-ans**. A **incomp-ans** tag is annotated for every incomplete or unfinished response. If the student did not answer a question at all, a **no-resp** tag is given. A difference in these dialogue error tags compared to the other error categories is that responses which are corrected with one of the dialogue tags, will not be “corrected”, and this information which will allow correction is usually missing. Table 3.12 gives some examples of the dialogue tags.

#### Table 3.12: Error annotation using dialogue error tags

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
<th>Error Tag Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where did you go on your last school holiday?</td>
<td>I am very happy.</td>
<td>unrel-ans</td>
</tr>
<tr>
<td>What did you like most about your school holiday?</td>
<td>I can reading and</td>
<td>incomp-ans</td>
</tr>
<tr>
<td>Describe your parents.</td>
<td>He is thin, friendly, and good parents</td>
<td>part-ans</td>
</tr>
<tr>
<td>Describe what your father does in his job.</td>
<td></td>
<td>no-resp</td>
</tr>
</tbody>
</table>
Table 3.13: Sentences annotated with the unclassifiable error tags

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
<th>Corrected Response</th>
<th>Error Tag Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tell me more about your country.</td>
<td>My country is peace, many building, minister, village.</td>
<td>My country is peaceful, has many buildings, ministers and villages.</td>
<td>unclsfid</td>
</tr>
<tr>
<td>What was the worst thing you did today?</td>
<td>I am not worst thing is play cycle.</td>
<td></td>
<td>unclsfid</td>
</tr>
</tbody>
</table>

Table 3.14: Grammatical Sentences

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
<th>Corrected Response</th>
<th>Error Tag Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tell me more about your city.</td>
<td>My city is a historical city.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What do you want to do next weekend?</td>
<td>I want to go to Pulau Pangkor.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.4.2.7 Unclassifiable Error Tags

The last error category is unclassifiable errors (unclsfid). A sentence is annotated as unclsfid if the sentence is marked with more than four error tags elected from the six error categories or the sentence’s errors do not fall into one of the categories. Table 3.13 gives some of examples.

3.4.2.8 Grammatical Responses

If a student answers a question using correct syntax, “Corrected Response” and “Error Tag” columns are left empty. This means no tags are provided for any grammatical responses as shown in Table 3.14.
Table 3.15: Ambiguous responses

<table>
<thead>
<tr>
<th>Question</th>
<th>Original Response</th>
<th>First Corrected Response</th>
<th>Second Corrected Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which country are you from?</td>
<td>I from Malaysia.</td>
<td>I am from Malaysia.</td>
<td>I come from Malaysia.</td>
</tr>
<tr>
<td>What will your parents do this evening?</td>
<td>They watching a television.</td>
<td>They will be watching television.</td>
<td>They will watch television.</td>
</tr>
<tr>
<td>What did you do last weekend?</td>
<td>I reading a comics.</td>
<td>I read comics.</td>
<td>I read a comic.</td>
</tr>
</tbody>
</table>

3.5 Errors Correction and Annotation

This section will explain how correction was done for every ungrammatical response. There were two steps taken in the process of correction: provision of a corrected sentence, and annotation of the error(s) by assigning relevant error tags.

3.5.1 Provision of Correct Models

If an annotator decided a response is ungrammatical, the annotator must provide its correct form. This sentence will be called the “model” sentence.

3.5.1.1 Ambiguous Utterances

For some ungrammatical responses, more than one ‘model’ sentence can be given. These responses are known as ambiguous responses. For each ambiguous response, a maximum of two model sentences are given. Table 3.15 shows some examples of ambiguous responses.

3.5.2 Annotating Tasks and Order of correction

After a corrected model was given, the annotator started doing annotation tasks. An incorrect response and its corresponding corrected models were compared. For each
discrepancy between them, a relevant error tag available in the error classification scheme (explained earlier in §3.4) was selected. The order of correction was from left to right based on the sequence of errors that occurred. Examples of the annotation process are shown in Table 3.16.

### 3.6 Inter-coder Reliability Tests

The annotation of error tags are tasks which assign appropriate error tags to certain errors. The work can be hard, complex, and confusing to annotators, especially when involving a large amount of data and more than one person doing the annotation. Different annotators (or coders) may have different understanding or views even though using a similar coding scheme. In order to ensure that all coders have an agreement during the annotation task, some sort of tests have to be applied to assess the strength of agreement, to ensure data reliability and validity of the scheme. These tests are good to measure agreement among coders. If good agreement can not be reached, invalid results will be produced.

As said by Krippendorff (2004), “researchers that wish to use hand-coded data, that is data in which items are labelled with categories, whether to support an empirical claim or to develop and test a computational model, need to show that such data are reliable.” The data are reliable if coders can be shown to agree on the categories assigned to units. Reliability is also a prerequisite for demonstrating the validity of the error classification scheme.

In the next section, I will explain three types of reliability tests in order of sophistication. The tests are percent agreement test, the kappa reliability test, and the alpha reliability test. The method I will use is the most sophisticated; §3.6.1 and 3.6.2 below can be understood as providing motivation for it. I will use the terms “inter-coder” and “inter-rater” interchangeably; and similarly the terms “agreement test” and “reliability test”.
<table>
<thead>
<tr>
<th>Question</th>
<th>Original Response</th>
<th>First Correction</th>
<th>Second Correction</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is your best friend's ambition?</td>
<td>She wants to be a fashion designer.</td>
<td>She wants to be a fashion designer.</td>
<td>She wants to be a fashion designer.</td>
<td>She wants to be a fashion designer.</td>
</tr>
<tr>
<td>What did your parents do last weekend?</td>
<td>They bought some groceries at the market.</td>
<td>They bought some groceries at the market.</td>
<td>They bought some groceries at the market.</td>
<td>They bought some groceries at the market.</td>
</tr>
<tr>
<td>Which country are you from?</td>
<td>I am from Malaysia.</td>
<td>I am from Malaysia.</td>
<td>I am from Malaysia.</td>
<td>I am from Malaysia.</td>
</tr>
<tr>
<td>What do your parents like doing?</td>
<td>They like going shopping.</td>
<td>They like going shopping.</td>
<td>They like going shopping.</td>
<td>They like going shopping.</td>
</tr>
<tr>
<td>What do you like doing?</td>
<td>I like camping, listening to music and playing games.</td>
<td>I like camping, listening to music and playing games.</td>
<td>I like camping, listening to music and playing games.</td>
<td>I like camping, listening to music and playing games.</td>
</tr>
<tr>
<td>What is your best friend going to do this evening?</td>
<td>She wants to reading a book.</td>
<td>She wants to read books.</td>
<td>She wants to read books.</td>
<td>She wants to read books.</td>
</tr>
</tbody>
</table>
Table 3.17: A example of inter-coder agreement table

<table>
<thead>
<tr>
<th></th>
<th>Coder 1</th>
<th></th>
<th>Coder 2</th>
<th></th>
<th></th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Statement</td>
<td>Request</td>
<td>Acknowledge</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Coder 1</td>
<td>30</td>
<td>5</td>
<td>10</td>
<td></td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>Coder 2</td>
<td>10</td>
<td>30</td>
<td>0</td>
<td></td>
<td></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0</td>
<td>30</td>
<td></td>
<td></td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>45</td>
<td>35</td>
<td>40</td>
<td></td>
<td></td>
<td>120</td>
</tr>
</tbody>
</table>

3.6.1 Percent Agreement Test

A percent agreement, $PA$ test was the earliest technique used to calculate an agreement level among coders. The agreement is also called observed agreement, $A_o$ which means the number of similar judgement the coders made when working on same data items. The percent agreement test is defined as the percentage of $A_o$. The $A_o$ is computed by

$$A_o = \frac{\text{the total of occurrence of same categories the coders assign}}{\text{the total number of items}}$$

Thus the percent agreement, $PA$ is calculated as

$$PA = A_o \times 100$$

Here is an example to measure percent agreement between two coders. The example was adopted from Allen and Core (1997), described in Artstein and Poesio (2008). See Carletta (1996) for further discussion on percent agreement test. Table 3.17 shows two coders, Coder 1 and Coder 2 assign Statement, Request or Acknowledge category on 120 utterances. Both coders agree on the Statement category, the Request category, and the Acknowledge category with the same total of 30 for each category.

The observed agreement,

$$A_o = \frac{30 + 30 + 30}{120} = 0.75$$

Therefore the percent agreement is

$$PA = 0.75 \times 100 = 75\%$$
The value can be interpreted as there is $75\%$ agreement between the two coders. The higher percentage shows the more perfect agreement. However, there are no standard measurement to consider high or low agreement.

The percent agreement test is not a satisfactory test to calculate inter-coder reliability for two reasons. Firstly, as pointed out by Scott (1955), the percent agreement test is biased especially when coders are just using a small number of categories. In other words, given two coding scheme for the same task, the one with fewer categories will get a higher percentage of reliability due to chance. Secondly, the test is not trusted because it does not correct for the distribution of items among categories. In other words, a higher percentage agreement is expected when one category is much more common than the other. In addition, Carletta (1996) argues that the percent agreement test is not efficient because it does not take into account the probability of chance agreement between raters. Instead, she suggests the Kappa reliability test to be adopted in measurement of reliability, especially in computational linguistics fields.

### 3.6.2 The Kappa Reliability Test

The Kappa ($\kappa$) reliability test considers two types of agreement: observed agreement, $A_o$ and expected agreement, $A_e$. The observed agreement is the proportion of items agreed by all coders. The expected agreement, $A_e$ is the level of agreement by all coders that can be attributed by chance. The formula of $\kappa$ is:

$$\kappa = \frac{A_o - A_e}{1 - A_e}$$

In the $\kappa$ test, perfect agreement has a value of 1 and full disagreement is 0. Different versions of $\kappa$ have been proposed especially in calculating an $A_e$ (see e.g. Scott, 1955; Cohen, 1960; Siegel and Castellan, 1988). See Eugenio and Glass (2004) for more information on different versions of $\kappa$ between Cohen and Siegel and Castellan (1988).

In my example, I choose Cohen’s formula to calculate $\kappa$. I would only consider calculating agreement between two coders. With reference to Table 3.17, the observed agreement, $A_o$ is is 0.75. This is the same for the $A_o$ in §3.6.1. The expected agreement,
Table 3.18: The $\kappa$ values and strength of agreement level

<table>
<thead>
<tr>
<th>$\kappa$ Values</th>
<th>Level of Agreement Strength</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0.00</td>
<td>Poor</td>
</tr>
<tr>
<td>0.00-0.20</td>
<td>Slight</td>
</tr>
<tr>
<td>0.21-0.40</td>
<td>Fair</td>
</tr>
<tr>
<td>0.41-0.60</td>
<td>Moderate</td>
</tr>
<tr>
<td>0.61-0.80</td>
<td>Substantial</td>
</tr>
<tr>
<td>0.81-1.00</td>
<td>Perfect</td>
</tr>
</tbody>
</table>

$A_e$ is measured by

$$A_e = P(c1, stat) \times P(c2, stat) + P(c1, req) \times P(c2, req) + P(c1, ack) \times P(c2, ack)$$

$$= \frac{45}{120} \times \frac{45}{120} + \frac{35}{120} \times \frac{40}{120} + \frac{40}{120} \times \frac{35}{120} = 0.335$$

which

$P(c1, stat)$ is the probability that Coder 1 assigns to the Statement category.

$P(c2, stat)$ is the probability that Coder 2 assigns to the Statement category.

$P(c1, req)$ is the probability that Coder 1 assigns to the Request category.

$P(c2, req)$ is the probability that Coder 2 assigns to the Request category.

$P(c1, ack)$ is the probability that Coder 1 assigns to the Acknowledge category.

$P(c2, ack)$ is the probability that Coder 2 assigns to the Acknowledge category.

Thus,

$$\kappa = \frac{A_o - A_e}{1 - A_e} = \frac{0.75 - 0.335}{1 - 0.335} = 0.624$$

In order to interpret the $\kappa$ value, Landis and Koch’s table was referred to. Table 3.18 depicts $\kappa$ strength level of agreement according to $\kappa$ values. The strength levels are scaled from perfect, substantial, moderate, fair, slight, and poor agreement. The perfect and substantial agreement show higher agreement (among coders) than slight and poor agreement. From the result of calculation, it could be interpreted that there is a substantial agreement between coders in the annotation tasks.
Note that $\kappa$ reliability only considers cases where raters fully agree or fully disagree. But in many cases, the raters can be said to agree to different levels on different cases. This could be to say that there is partial agreement (not full agreement) among the coders. If partial agreement is required to be counted, the $\kappa$ is not suitable as lower results will be obtained. Other reliability tests, e.g. the Alpha test, should be considered instead.

### 3.6.3 The Alpha Reliability test

The Alpha ($\alpha$) test was invented by Krippendorff (Krippendorff, 1980) to overcome weaknesses in the $\kappa$ reliability test. $\alpha$ takes into account various degrees of agreement to calculate inter-coder reliability. While $\kappa$ computes $A_o$ and $A_e$, $\alpha$ measures observed disagreement, $D_o$ and expected disagreement, $D_e$ instead. $D_o$ is the proportion of items on which all coders disagree while $D_e$ is the probability of all coders disagreeing on any categories due to chance. The formula of the $\alpha$ reliability test is:

$$\alpha = 1 - \frac{D_o}{D_e}$$

When $\alpha=1$, it indicates perfect agreement which means $D_e=0$. The various degrees of agreement mean situations occurred when the coders agree partially on their counterparts’ judgement. Full disagreement value is 0, so ranges of value which indicate degrees of agreement are between 0 to 1. The value is known as a distance metric, $\delta$. The value of $\delta$ indicates the degree of agreement. Logically, $\delta=0$ for full agreement while $\delta=1$ for full disagreement. For any partial agreement among coders, $0.0<\delta<1.0$.

A detailed discussion about the distance metric is in Krippendorff (2004). Research studies that apply $\alpha$ reliability test are Passonneau (2004) and Passonneau (2006). In addition, Passonneau invents her own $\delta$ values called MASI (Measuring Agreement on Set-valued Items) in her research for calculating inter-coder reliability of semantic and pragmatic annotation project.

Now, let’s go further on how $D_o$ and $D_e$ are calculated. The $D_o$ is obtained by using a formula of

$$D_o = \frac{1}{i(c-1)} \sum_i \sum_{c_1} \sum_{c_2} n_{ikc_1} n_{ikc_2} \delta_{kc_1kc_2}$$
where

\[ c \] is the number of coders, and \( i \) is the number of items.

\[ c1 \] and \( c2 \) identifies Coder 1 and Coder 2 respectively.

\[ k \] is a category assigned by the coders.

\( n_{ik} \) represents the number of items \( i \) given the category \( k \) by any coder.

\( \delta_{k1,k2} \) determines a distance value between categories assigned by each coder.

\[ D_e \] is computed by using a formula of

\[
D_e = \frac{1}{ic(i-1)} \sum_{c1} \sum_{c2} n_{k1} n_{k2} \delta_{k1,k2}
\]

where

\( n_k \) is the total number of items. The category \( k \) is assigned to any item by any coders.

As an example of calculating \( \alpha \), I used the same data in Table 3.17. There is a distinct different between Statement and Request but Acknowledge is somehow said to have closer relationship to Statement. It means if one coder assigns Acknowledge and another coder assigns Statement on a same item, partial agreement is considered. The distance metric, \( \delta \) value which shows the weight of agreement for each pair of categories is outlined below.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Request</th>
<th>Acknowledge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statement</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Request</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>0.5</td>
<td>1</td>
</tr>
</tbody>
</table>

The observed disagreement, \( D_o \) is calculated by multiplying each cell in Table 3.17 by its respective \( \delta \) as provided above, summing up all each result of the cells, and dividing the total by the number of items.

\[
D_o = \frac{30 \times 0 + 5 \times 1 + 10 \times 0.5 + 10 \times 1 + 30 \times 0 + 0 \times 1 + 5 \times 0.5 + 0 \times 1 + 30 \times 0}{120} = \frac{22.5}{120} = 0.1875
\]
Table 3.19: The range of $\alpha$ values and its interpretation

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0.67 &lt; \alpha &lt; 0.8$</td>
<td>Tentative conclusions to be drawn</td>
</tr>
<tr>
<td>$\alpha &gt; 0.8$</td>
<td>Good reliability</td>
</tr>
</tbody>
</table>

The calculation of $D_e$ is the sum of the expected disagreement for each category pair multiplied by its distance metric. Below is the calculation:

$$D_e = \frac{(45 + 45) \times (45 + 45)}{2 \times 120 \times (2 \times 120 - 1)} \times 0 + \frac{(35 + 40) \times (45 + 45)}{2 \times 120 \times (2 \times 120 - 1)} \times 1 + \frac{(40 + 35) \times (45 + 45)}{2 \times 120 \times (2 \times 120 - 1)} \times 0.5 + \frac{(45 + 45) \times (35 + 40)}{2 \times 120 \times (2 \times 120 - 1)} \times 1 + \frac{(35 + 40) \times (35 + 40)}{2 \times 120 \times (2 \times 120 - 1)} \times 0 + \frac{(40 + 35) \times (35 + 40)}{2 \times 120 \times (2 \times 120 - 1)} \times 1 + \frac{(35 + 40) \times (40 + 35)}{2 \times 120 \times (2 \times 120 - 1)} \times 0.5 + \frac{(40 + 35) \times (40 + 35)}{2 \times 120 \times (2 \times 120 - 1)} \times 0$$

$$= 0.5492$$

Thus, given by the formula $1 - \frac{D_o}{D_e}$, the value of $\alpha$ is $1 - \frac{0.1875}{0.5492} = 0.6586$. In order to interpret the $\alpha$ value, Krippendorff has his own interpretation as illustrated in Table 3.19. Krippendorff’s $\alpha$ is more stringent than $\kappa$ interpretation using Landis and Koch’s table. Although Krippendorff does not include any interpretation for $\alpha < 0.67$, from the result I obtained, I would interpret it as weak reliability between the coders.

Further discussions and examples of measuring $\alpha$, and various versions of $\kappa$ reliability between two or more coders can be found in Artstein and Poesio (2008).

### 3.7 Assessing Inter-Coder Agreement

In the computational linguistics area, it is only recently that annotation projects have included a formal test of inter coder agreement (e.g., Passonneau, 2006, 2004). I applied the reliability tests in order to ensure the reliability of my data and the validity of my error classification scheme. I am not interested in assessing the reliability of the process of assigning correct model answers. That’s known to be quite unreliable. I am interested in assessing the reliability of error annotation using the error classification scheme, given an incorrect sentence and a proposed correct sentence.

In response to the unreliability in measuring agreement for the provision of correct answers, Michaud (2002) notes in her thesis: “...even native speakers’ judgement may differ ...”. She reports many such differences when two coders corrected a subset
of writing samples completed by deaf students of American Sign Language. Other research conducted by Voorhees and Tice (2000) was to investigate judgement of multiple assessors when assessing users’ responses in their question answering system, TREC. Results show different assessors often have different opinions about what constitutes an acceptable response. For example, for the question “When did Nixon die?”, some assessors accept “April 22” as a correct answer, while others require the year as well. On average, 94% of users’ responses are agreed by all assessors. This shows different people have different opinions, perceptions and preferences. This is the reason for why I do not assess the agreement of correct answers provided by coders. I want to evaluate the reliability of error annotation done by the coders, instead.

3.7.1 Coders for the Reliability Tests

Two coders were involved in the annotation tasks: myself and my primary supervisor. I randomly selected four students’ questionnaires which gave a total of 180 utterances (4 × 45 questions). We both worked on the same subset of data. Both coders read each question and its corresponding response. The coders followed the annotation protocol, as explained earlier in §3.5.

3.7.2 Prior to the Reliability Tests Process

After the tasks were accomplished, the two coders met together to discuss the model answers they both provided for each ungrammatical response. This is to ensure that error annotation was applied based on a model answer which was correct and agreed by both coders. Our reasoning for this was one coder is a English native speaker and the other one speaks English as her second language. In the following text, I refer to the native speaker as Coder 1 and myself as Coder 2.

Discussion and negotiation were done for the following scenarios. There were some cases where Coder 2 provided a model answer which is incorrect, and judged an ungrammatical response as a grammatical one or vice versa. In such scenario, the model answer given by Coder 1 was taken. Yet, there are some responses consistent with information about Malaysia, such as places and cultures which were beyond Coder 1’s
knowledge. Coder 2 knows about Malaysian places and cultures. In this case, Coder 2 explained to Coder 1 about the information. Other cases occurred where different model answers were provided by both coders which both agreed were valid. In these cases, a maximum of two model answers were accepted for the ungrammatical response.

After the discussion was done and the set of model answers was agreed, both coders separately did the error annotation for these model answers. This time, both coders did the tasks independently, without any discussion between them. When the tasks were accomplished, the reliability tests began.

### 3.7.3 Measuring Inter-coder Reliability

This section will explain how the reliability tests were applied. Before describing in detail how inter-coder agreement was measured, I will describe the procedures for judging inter-coder agreement. The first one is a pairwise comparisons method to match error tags. This is followed by a comparison of individual error tags procedure.

Firstly, agreement was checked based on pairwise comparisons of the error tags. The comparisons were done by matching two error tags with the same location. The first error tag assigned by Coder 1 was compared with the first error tag assigned by Coder 2, and so on. For instance, in Table 3.20, agreement was achieved for Tag 1 and Tag 2. No agreement was considered for Tag 3 and Tag 4 even though both coders assigned the same errors tags but in opposite location. In Agreement row, Yes means full agreement and No means full disagreement.

Table 3.21 shows an example of coders’ annotation. Coder 2 does not include a tag, T2 which causes disagreement beginning from Tag 2 until Tag 4. This may be due to the human mistake of slipping tags. The problem can be solved by rearranging
Table 3.21: Sequence of error tags.

<table>
<thead>
<tr>
<th></th>
<th>Tag 1</th>
<th>Tag 2</th>
<th>Tag 3</th>
<th>Tag 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder 1</td>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>Coder 2</td>
<td>T1</td>
<td>T3</td>
<td>T4</td>
<td></td>
</tr>
<tr>
<td>Agreement</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3.22: Error tags realignment.

<table>
<thead>
<tr>
<th></th>
<th>Tag 1</th>
<th>Tag 2</th>
<th>Tag 3</th>
<th>Tag 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder 1</td>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>Coder 2</td>
<td>T1</td>
<td></td>
<td>T3</td>
<td>T4</td>
</tr>
</tbody>
</table>

the sequence of tags. For example, Coder 2’s T3 and T4 tags are shuffled forward one place ahead. The result of rearrangements is depicted in Table 3.22.

Tag misalignment is also an issue looked by Michaud (2002). Michaud applies a realignment algorithm (Smith and Waterman, 1981), giving penalties for any realignment which were needed. There are two reasons why I didn’t perform any alignment tasks. Firstly, if realignment is required, the question of whose error tags should be selected arises. Again in Table 3.21, if Coder 1 is chosen, T3 and T4 are moved backward one place. As a result, T2 is deleted and this causes incomparable between Coder 1’s error annotation and the occurrence of errors in the respective utterance. Nevertheless, such problems can be avoided if only Coder 2’s T3 and T4 tags are brought forward one location.

The second reason is when both coders have assigned the maximum numbers of error tags. As referred in Table 3.23, if T4 and T5 of Coder 2 are shuffled forward one location, the total number of error tags exceeds four. If T5 is removed, again the error annotation does not tally with the occurrence of errors in the utterance. Therefore, I decided not to apply tag realignment. Furthermore, only five cases occurred in my error annotation.

Now I explain the second procedure of inter-coder agreement judgement. It involves the matching of individual error tags. As stated before, each error tag is represented in a predicate form. Each predicate has up to two arguments. There are 15 predicates.
to choose from and among them, there are two predicates with one argument and three predicates with two arguments. More than twenty linguistic forms are available for identifying the arguments (see Table 3.10 on page 84 and Tense Errors section on page 81). Due to many predicate types and their respective arguments, I decided to assess the inter-coder agreement in two levels, as outlined below:

**Level 1:** The comparison of individual error tag predicates only without looking at their arguments.

**Level 2:** For each agreed pair of predicate, arguments of the pair are compared.

In Level 1, the comparison was only based on the predicate of error tags. Full agreement was considered if both predicates were matched if their arguments was different, as shown in Table 3.24.

When the predicates were not matched, this was assessed as full disagreement as shown in Table 3.25. Another case of disagreement was when both coders annotated the same error tags but the tags were ordered in an opposite location as shown in Table 3.26.

One more disagreement case was when a coder slipped one tag, at least. An example is as shown in Table 3.27. Full agreement was judged for Tag 1 and Tag 2 but not for Tag 3 and Tag 4. Even though Coder 2 assigned three tags only, matching was still done in the Tag 4 column.

---

**Table 3.23: Limited number of error tags.**

<table>
<thead>
<tr>
<th></th>
<th>Tag 1</th>
<th>Tag 2</th>
<th>Tag 3</th>
<th>Tag 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder 1</td>
<td>T1</td>
<td>T2</td>
<td>T3</td>
<td>T4</td>
</tr>
<tr>
<td>Coder 2</td>
<td>T1</td>
<td>T2</td>
<td>T4</td>
<td>T5</td>
</tr>
</tbody>
</table>

**Table 3.24: Equivalent predicates in a same sequence order.**

<table>
<thead>
<tr>
<th></th>
<th>Tag 1</th>
<th>Tag 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder 1</td>
<td>sva(X)</td>
<td>ins(X)</td>
</tr>
<tr>
<td>Coder 2</td>
<td>sva(X)</td>
<td>ins(X)</td>
</tr>
<tr>
<td>Agreement</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

---

**Table 3.25:** Equivalent predicates in a same sequence order.
Table 3.25: Different error tags annotation.

<table>
<thead>
<tr>
<th>Tag 1</th>
<th>Coder 1</th>
<th>Coder 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>det-n-ag</td>
<td></td>
<td>del(X)</td>
</tr>
<tr>
<td>Agreement</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3.26: Equivalent tags but in different order.

<table>
<thead>
<tr>
<th>Tag 1</th>
<th>Tag 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>del(X)</td>
<td>ins(X)</td>
</tr>
<tr>
<td></td>
<td>del(X)</td>
</tr>
<tr>
<td>Agreement</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 3.27: Missing tags

<table>
<thead>
<tr>
<th>Tag 1</th>
<th>Tag 2</th>
<th>Tag 3</th>
<th>Tag 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>del(X)</td>
<td>del(X)</td>
<td>sva(X)</td>
<td>ins(X)</td>
</tr>
<tr>
<td>del(X)</td>
<td>del(X)</td>
<td>ins(X)</td>
<td></td>
</tr>
<tr>
<td>Agreement</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 3.28: Agreement in arguments of predicate

<table>
<thead>
<tr>
<th></th>
<th>Tag 1</th>
<th>Tag 2</th>
<th>Tag 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coder 1</td>
<td>sva(have)</td>
<td>ins(noun)</td>
<td>del(noun)</td>
</tr>
<tr>
<td>Coder 2</td>
<td>sva(have)</td>
<td>ins(noun)</td>
<td>del(adj)</td>
</tr>
<tr>
<td>Agreement</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

I now describe the Level 2 of agreement test. In this level, I calculated the agreement of arguments for each predicate that has similar judgement by both coders. The error tag predicates which have arguments are \( \text{del}(X) \), \( \text{ins}(X) \), \( \text{sva}(X) \), \( \text{subst-with}(X,Y) \), \( \text{tense-error}(A,B) \), and \( \text{transp}(X,Y) \). An example of full agreement and full disagreement is shown in Table 3.28.

During this measurement, some cases of partial agreement occurred between two coders. Refer to an example below:

Coder 1  \( \text{ins}(\text{will}) \)  
Coder 2  \( \text{ins}(\text{modal-aux}) \)

Coder 1 tagged \( \text{ins}(\text{will}) \) and Coder 2 tagged \( \text{ins}(\text{modal-aux}) \). Partial agreement was considered because \( \text{will} \) is categorised as one type of modal auxiliary. Despite both arguments being included in the error classification scheme, \( \text{modal-aux} \) is referred to other modal auxiliary types such as \( \text{can} \) and \( \text{may} \).

Another case of partial agreement can occur in \( \text{tense-err}(X,Y) \) error tags. For instance,

Coder 1  \( \text{tense-err}(\text{past},\text{pres}) \)  
Coder 2  \( \text{tense-err}(\text{past},\text{progr}) \)

Since one of the arguments match, this was assessed as a case of partial agreement.

To measure inter-rater agreement for my annotation scheme, I applied the \( \alpha \) reliability test. \( \alpha \) reliability can be applied because it caters for different levels of agreement as noted earlier. The weight is indicated by a distance metric, \( \delta \). In this study, I assigned \( \delta=0.5 \) for the above two types of partial agreement cases. For full agreement,
Table 3.29: Distance metric used in the ($\alpha$) test

<table>
<thead>
<tr>
<th>Distance Metric Value</th>
<th>Descriptions</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If coder1’s tag == coder2’s tag</td>
<td>Coder1: <code>ins(noun)</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coder2: <code>ins(noun)</code></td>
</tr>
<tr>
<td>0.5</td>
<td>1) If coder1’s and coder2’s tag is <code>modal-aux</code> and <code>will</code> or vice versa <code>inf-mrkr-to</code> and <code>to</code> or vice versa 2) If 1st argument in coder1’s and coder2’s tag or 2nd argument in coder1’s and coder2’s tag is similar</td>
<td>Coder1: <code>del(modal-aux)</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coder2: <code>del(will)</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coder1: <code>tense-err(progr,pres)</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coder2: <code>tense-err(progr,inf)</code></td>
</tr>
<tr>
<td>1</td>
<td>If coder1’s tag $\neq$ coder2’s tag</td>
<td>Coder1: <code>del(be)</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coder2: <code>subst-with(be,will)</code></td>
</tr>
</tbody>
</table>

$\delta=0$, and full disagreement, $\delta=1$. A summary of distance metric values for the $\alpha$ test is shown in Table 3.29.

3.7.4 Results of the reliability tests

The reliability tests were calculated using the script developed by Lippincott (2000). The script was developed using the Python programming language. Table 3.30 shows the overall results of the $\kappa$ and $\alpha$ tests.

In order to interpret the $\kappa$ values, I referred to $\kappa$ strength level of agreement according to $\kappa$ values (Landis and Koch, 1977), as illustrated in Table 3.18 on page 93. Also, Krippendorff (2004)’s policies on interpreting $\alpha$ values, as outlined in Table 3.19 on page 96, was used to describe $\alpha$ agreement level.

The $\kappa$ test results show there is a substantial agreement between two coders in the judgement of all error tags predicate, and the judgement of arguments of `ins(X)` and `tense-err(A, B)` error tags. Tentative conclusions need to be drawn for the $\alpha$ values for those three values. However, there is evidence of a perfect agreement ($\kappa$ value)
Table 3.30: Results of agreement tests

<table>
<thead>
<tr>
<th>Agreement on</th>
<th>The $\alpha$ Test</th>
<th>The $\kappa$ Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>All predicates of error tags</td>
<td>0.79</td>
<td>0.79</td>
</tr>
<tr>
<td>The argument of the $\text{sva}(X)$</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>The argument of the $\text{del}(X)$</td>
<td>0.94</td>
<td>0.91</td>
</tr>
<tr>
<td>The argument of the $\text{subst-with}(X,Y)$</td>
<td>0.97</td>
<td>0.95</td>
</tr>
<tr>
<td>The argument of the $\text{ins}(X)$</td>
<td>0.78</td>
<td>0.66</td>
</tr>
<tr>
<td>The argument of the $\text{tense-err}(A,B)$</td>
<td>0.72</td>
<td>0.71</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.87</strong></td>
<td><strong>0.84</strong></td>
</tr>
</tbody>
</table>

or a good reliability ($\alpha$ value) between two coders in the judgement of arguments of $\text{sva}(X)$, $\text{del}(X)$, and $\text{subst-with}(X,Y)$.

Overall, the $\alpha$ and $\kappa$ test average results, 0.87 and 0.84 respectively, which indicates a good reliability (for $\alpha$ scale) or a perfect agreement (for $\kappa$ scale) are found between the coders. It is shown that both coders have reached a good/perfect agreement and the error classification scheme for error annotation is valid. Even though Krippendorff considers 0.80 is a minimum $\alpha$ value to be accepted, we (my two supervisors and I) are satisfied with our reliability results. The annotation of an error tag is considered to be quite a difficult task, especially when there are many types of error tags to be referred to. As a result, we decided to carry on with the error annotation process tasks based on the error classification scheme for the remaining data.

I continued correcting and annotating the remaining data. Firstly, I provided a maximum of two model answers for every ungrammatical response. My two supervisors helped me by checking the model answers which I provided. Each of them checked half of the corpus data. After that, I did the error annotation. In the following section, I will explain the results of data analysis and how I interpret them.
3.8 Results of the Corpus Data Analysis

As described earlier in §3.2, 7785 responses were checked. Error tags annotation was done for every ungrammatical response. Once the annotation tasks were completely accomplished, the data was analysed. This section explains about the analysis results for the following questions:

1. What is the most common error made by all students?

2. How does performance of the students change across form levels?

Results for the first question will be described in §3.8.1. Then §3.8.2 presents findings yielded from the second question.

3.8.1 Most Common Errors

The distribution of error categories across all student groups is depicted in Figure 3.1. The most common error which occurred is the error of missing a word in a sentence which is represented by the \texttt{ins(X)} error tag. Its percentage is 31% of total numbers of errors.

The \texttt{ins(X)} error category means the students are missing a word in their sentence. Since this kind of error has an argument covers words or linguistic forms, as shown in Table 3.10 (page 68), we can also question what is the most common category of the absent word. Data about this question is depicted in Figure 3.2.

In Figure 3.2, the word categories which are most frequently missing in the students’ responses are a determiner, the copula \textit{be}, the auxiliary \textit{will}, pronouns, and possessive noun morphology. These categories are also known as closed-class word types: the number of words in each of these categories are limited. A determiner word category represents an indefinite determiner (\textit{indef-det}) and a definite determiner (\textit{def-def}). Examples of determiners, the copula \textit{be}, pronouns and possessive noun morphology can be found in Table 3.10 (page 84).
Figure 3.1: Distribution of percentage of each error category

Figure 3.2: Percentage of a word missing in the $\text{ins}(X)$ error tag category
3.8.1.1 **Omission of determiner and the copula \textit{be}**

The most commonly omitted word category is the type determiner which contributes 23% of the overall percentage (in Figure 3.2). The copula \textit{be} comes up second in the list. A study of Malaysian EFL students conducted by Maros \textit{et al.} (2007) also showed that the incorrect use of these two categories are the most frequent errors made by their participating students. Examples of such errors are shown below:\(^4\):

[Example 1 - \texttt{ins(indef-det)}]: I want to be * lawyer.

[Example 2 - \texttt{ins(def-det)}]: We go to * library together.

[Example 3 - \texttt{ins(be)}]: My name * Nora.

The absence of determiners and copulas in the students’ sentences is perhaps due to a transfer rule error. (There are no determiners or copulas in Malay - see e.g. Hassan (1993).) To investigate this possibility I include Figure 3.3 which depicts the percentage of a word missing in the \texttt{ins(X)} error tag category for the Sch1 school only, where the large majority of students in Sch1 are Malays. In fact, the figure shows a similar distribution of determiner and copula insertion errors as in the general population. (However, it should be borne in mind that Chinese also lacks the copula in many contexts (Chan, 2004) and also lacks many classes of determiner [see e.g. Alexiadou, Haegeman, and Stavrou (2007)]).

Determiners in the English language form part of a noun phrase. They occur in front of a head noun of the noun phrase. In the Malay grammar, a head noun does not require to be preceded with a determiner. Table 3.31 shows some examples of sentences which the students failed to insert the determiner.

The copula \textit{be} is the second highest missing word in the students’ sentences and contributes 18% of the total percentage. There is no copula in Malay. So this could be another transfer error. Due to the various forms of English copula, which vary according to the subject and tenses, and its non-existence in the Malay language, Maros \textit{et al.} (2007) found the copula \textit{be} is most problematic to the students. Table 3.32 shows some examples of sentences which the students failed to insert the copula \textit{be}.

\(^4\)Symbols * show either a \texttt{determiner} or a \texttt{be} word category should be inserted.
Figure 3.3: Percentage of a word missing in the $\text{ins}(X)$ error tag category for the Sch1 school

<table>
<thead>
<tr>
<th>Table 3.31: The absence of determiners</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Erroneous form</strong></td>
</tr>
<tr>
<td>English: I want to be * lawyer.</td>
</tr>
<tr>
<td>Malay: Saya ingin menjadi peguam.</td>
</tr>
<tr>
<td>English: We go to * library together.</td>
</tr>
<tr>
<td>Malay: Kami pergi ke perpustakaan bersama-sama.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3.32: The absence of the copula be</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Erroneous form</strong></td>
</tr>
<tr>
<td>English: My name * Nora.</td>
</tr>
<tr>
<td>Malay: Nama saya Nora.</td>
</tr>
<tr>
<td>English: I * from Melaka</td>
</tr>
<tr>
<td>Malay: Saya dari Melaka.</td>
</tr>
</tbody>
</table>
3.8.1.2 Incorrect use of Tense

The second highest common error occurred, as shown in Figure 3.1 is $\text{tense-err}(A,B)$. As this type of error category has two arguments, Figure 3.4 shows the distribution of its arguments. It is shown that 50% of the overall errors involved the use of present tense words instead of past tense words. These errors occurred in responses for Past Tense questions. The second highest errors are $\text{tense-err}(\text{progr, inf})$. This means the students incorrectly use a progressive tense word instead of an infinitive tense word. These errors often happened in response to a question, “What do(es) you/your best friend/your parents like doing?” Again, the prevalence of this error most likely is due to grammatical transfer from L1. Figure 3.5 shows the distribution of this error committed by all students in Sch1, the predominantly Malay school. The pattern of results is very similar to those shown in Figure 3.4. Again, it should be borne in mind that Mandarin also lacks tense markings on verbs (see e.g. Shi (2002)), so the incidence of tense errors in the general population could be due to a combination of similar transfer effects. Tables 3.33 and 3.34 show some examples of students’ sentences using the incorrect tense forms.

However, the above mentioned reason is only partly explaining the difficulty. The students have learnt all basic English tenses such as simple present tense, infinitive tense, past tense, progressive/continuous tense, and perfect tense. Another explana-
Figure 3.5: Percentage of tense error category distribution for the Sch1 school only

Table 3.33: Present and Past tenses

<table>
<thead>
<tr>
<th>Present Tense</th>
<th>Past Tense</th>
</tr>
</thead>
<tbody>
<tr>
<td>English: I go to Melaka today.</td>
<td>I went to Melaka last week.</td>
</tr>
<tr>
<td>Malay: Saya pergi ke Melaka hari ini.</td>
<td>Malay: Saya pergi ke Melaka minggu lepas.</td>
</tr>
<tr>
<td>English: I am born at Hospital Melaka.</td>
<td>I was born at Hospital Melaka.</td>
</tr>
<tr>
<td>Malay: Saya dilahirkan di Hospital Melaka.</td>
<td>Malay: Saya dilahirkan di Hospital Melaka.</td>
</tr>
</tbody>
</table>

Table 3.34: Progressive and Infinitive tenses

<table>
<thead>
<tr>
<th>Incorrect Tense (Progressive)</th>
<th>Correct Tense (Infinitive)</th>
</tr>
</thead>
<tbody>
<tr>
<td>English: They like to gardening.</td>
<td>They like to garden.</td>
</tr>
<tr>
<td>Malay: Mereka suka berkebun.</td>
<td></td>
</tr>
<tr>
<td>English: He * going to the field to playing football.</td>
<td>He is going to the field to play football.</td>
</tr>
<tr>
<td>Malay: Dia akan ke padang untuk bermain bolasepak.</td>
<td></td>
</tr>
</tbody>
</table>
tions could be that the students may be not aware of the tense form used in a question or still possess limited knowledge of tenses. An example could be, in response to the question “Where did you go on your last school holiday?”, a student just follows the verb word “go” for example “I go to Melaka.”

3.8.1.3 Discussion

The information of most common errors also will inform English language teachers to pay more attention to this area of grammar. The most common errors made are an absence of determiners and of the copula be, and incorrect tense used in students’ sentences. Determiners, the copula be, and tense forms do not exist in students’ L1, therefore L1 interference may be one of reasons why the students made this particular error so frequently. Mohideen (1996) also mentions students’ L1 interference as one of reasons why the students are making errors during learning of English. Jalaluddin et al. (2008); Maros et al. (2007); Mohideen (1996) suggest, in order to counter the problem, the teachers should do a comparative analysis between English and the students’ L1.

3.8.2 Longitudinal Comparison

In this section, I will investigate whether students make progress as they move up to a higher form level. For example, students in Form 3 should possess more grammatical knowledge than Form 2, and Form 2’s students should be better than students in Form 1. By right the higher form level students should make fewer mistakes than the lower form students. This information could be very useful in providing detailed quantitative information to language teachers about how their students progress over time. Therefore I applied statistical tests to compare the Form levels for all schools.

Although a huge amount of data was collected, I faced difficulty in making this longitudinal comparison. There was a lack of control on the number of students and different types of class streams. (Refer to Table 3.1 in page 68). Despite that all students in Sch2 are from the same class stream, the total numbers of students in each class are too small to be compared (less than twenty students). Yet, Sch3 has a small number of students in Form 1 and Form 3 and the class stream of all students in Sch3
are mixed. Therefore I decided to focus on the students in Form 1 and Form 3 from the Sch1 school because both groups had relatively similar numbers of students and were in the same class streams.

3.8.2.1 General Differences between Form 1 and Form 3 of Sch1 school students

There were about 34 Form 1 students with a total of 1530 utterances and 25 Form 3 students with 1125 total utterances. The grand total of all utterances is 2655 utterances. The Form 1 and Form 3 students are labelled as Group 1 and Group 2 respectively. “Form 1”, “Group 1” and “Form 3”, “Group 2” will be used interchangeably. Figure 3.6 shows an average of errors with standard error bars, made per student in each group. For 45 questions given, on average every student in Form 1 committed 55 errors whereas every Form 3 student committed 32 errors in their responses.

To test whether this difference is significant, a one-tailed Student’s \( t \) test was performed. The result yielded is \( t = 6.36, df = 57, p < 0.05 \), which means the number of errors made by students in Group 2 is significantly lower than the number made by students in Group 1. Moreover, on average Group 2 produces more words than Group 1 as depicted in Figure 3.7. This shows that Group 2 used more words in their sentence and made fewer mistakes. Therefore at a broad-brush level, the answer to my second research question posed in §3.8 is that students in higher form levels do indeed perform better than the students in lower levels.

However, my data permit me to consider inter Form differences at a much greater level of detail. We can examine how the distribution of students’ errors changes from Group 1 to Group 2. The distributions of error categories for Group 1 and Group 2 are depicted in Figure 3.8. This figure presents useful information about how patterns of each error category for Group 1 and Group 2 change over time. As shown here, the students in Group 2 have made a lot of improvement especially across the board, in making fewer mistakes for the \( \text{ins}(X) \), \( \text{tense-err}(A,B) \) and \( \text{del}(X) \) error categories. However, to prove the results mathematically, more statistical tests need

---

5 Standard error bars are referred to the division of a standard deviation of total errors and a square root of total number of students in each group.
Figure 3.6: Average of total errors committed per student (with standard error bars)

Figure 3.7: Average of total words produced per student (with standard error bars)
Figure 3.8: Average of total each error category done per student for each group

to be performed to check the difference between the two groups. Next, I will explain the procedure I followed in order to check for a significant difference for each error category.

3.8.2.2 Difference in the distribution of errors between Form 1 and Form 3

In order to examine the performance of Group 1 and Group 2, statistical tests were applied to each error category. I followed the following steps in order to perform the tests.

1. A pre-requisite to apply a Student’s $t$ test, sample data must come from the normal distribution. Therefore, I performed the Shapiro-Wilks normality test on each distribution of both groups for each error category.

2. If the distribution of both groups was from the normal distribution, next I applied the Fisher’s $F$ test to compare both variances of two distributions.
(a) If both distributions were homogeneous, I applied a one-tailed Student’s \( t \) test with an equal variance as one of its parameter.

(b) Otherwise, the one-tailed Student’s \( t \) test with non-equal variance parameter was performed.

3. If only either one of the distributions was not or both distributions were not from the normal distribution, I performed the Wilcoxon’s rank test, which does not require an assumption of normality.

The steps are summarised in a flowchart that is depicted in Figure 3.9. All tests were calculated by using the R software (Crawley, 2007). The results of significant tests are outlined in Table 3.35. The \( t \) and \( W \) symbols indicate that the Student’s \( t \) test and the Wilcoxon’s rank test were applied. In the following paragraph, I will explain in detail the statistical test results for the first five error categories listed in Table 3.35. I won’t further explain the remaining error categories as the average value for both groups is slightly small.

The first four error categories (\( \text{ins}(X) \), \( \text{tense-err}(A,B) \), \( \text{del}(X) \) and \( \text{subst-with}(X,Y) \)) will be discussed here. The results show those categories for Group 1 differed significantly from Group 2. However, these categories have arguments, so we should also consider comparisons between groups for the most common patterns of argument. Further analyses were done in which I only focused on the first three highest types of arguments of those selected error categories. The results are presented in Table 3.36.

The table shows Group 2 has made significant progress in committing fewer errors than Group 1 in \( \text{ins}(\text{be}) \), \( \text{ins}(\text{will}) \), \( \text{tense-err}(\text{pres,past}) \), \( \text{tense-err}(\text{progr,inf}) \), \( \text{tense-err}(\text{progr,past}) \), \( \text{del}(\text{be}) \), and \( \text{del}(\text{verb}) \) errors. On the other hand, there is no statistical difference in the \( \text{ins}(\text{det}) \), \( \text{del}(\text{det}) \), \( \text{subst-with}(\text{verb,verb}) \), \( \text{subst-with}(\text{noun,noun}) \), and \( \text{subst-with}(\text{at,in}) \) errors categories. What can explain the insignificant results in the \( \text{ins}(\text{det}) \), and \( \text{del}(\text{det}) \) error categories? These show that Group 1 and Group 2 are still having difficulties with the use of determiners. Group 2 does not perform well in the use of determiners even though it should possess more grammar knowledge than Group 1. As mentioned earlier in §3.8.1 above, this situation
Figure 3.9: A flowchart to show which significant tests are performed
Table 3.35: Results of Statistical Significant Tests for each error category

<table>
<thead>
<tr>
<th>Error Category</th>
<th>Statistical Tests</th>
<th>Statistically significant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ins($X$)</td>
<td>$t=5.6562$, $df=52.867$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>tense-err($A,B$)</td>
<td>$t=5.339$, $df=57$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>del($X$)</td>
<td>$W=626$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>subst-with($X,Y$)</td>
<td>$t=2.0479$, $df=57$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>sva($X$)</td>
<td>$t=0.1934$, $df=57$, $p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>spell-err</td>
<td>$W=561.5$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>noun-num-err</td>
<td>$W=532.5$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>det-n-ag</td>
<td>$W=560.5$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>unrel-ans</td>
<td>$W=649.5$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>unclsfd</td>
<td>$W=544.5$, $p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>vocab-err</td>
<td>$W=496.5$, $p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>transp($X$, $Y$)</td>
<td>$W=448.5$, $p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>incom-p-ans</td>
<td>$W=464.5$, $p&gt;0.05$</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 3.36: Results of Statistical Significance Tests for arguments of the ins($X$), tense-err($A,B$), del($X$) and subst-with($X,Y$) error categories

<table>
<thead>
<tr>
<th>Error Category</th>
<th>Statistical Tests</th>
<th>Statistically significant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ins($det$)</td>
<td>$W=447, p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>ins($be$)</td>
<td>$W=745.5, p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>ins($will$)</td>
<td>$W=737.5, p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>tense-err($pres,past$)</td>
<td>$W=661, p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>tense-err($progr,inf$)</td>
<td>$W=669.5, p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>tense-err($progr,past$)</td>
<td>$W=639, p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>del($det$)</td>
<td>$W=459.5, p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>del($be$)</td>
<td>$W=571, p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>del($verb$)</td>
<td>$W=527.5, p&lt;0.05$</td>
<td>Yes</td>
</tr>
<tr>
<td>subst-with($verb,verb$)</td>
<td>$W=457.5, p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>subst-with($noun,noun$)</td>
<td>$W=416, p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>subst-with($at,in$)</td>
<td>$W=348, p&gt;0.05$</td>
<td>No</td>
</tr>
</tbody>
</table>
Table 3.37: Subject-verb agreement error

<table>
<thead>
<tr>
<th>Erroneous form</th>
<th>Correct form</th>
</tr>
</thead>
<tbody>
<tr>
<td>English: I <em>has</em> one sister.</td>
<td>I have one sister.</td>
</tr>
<tr>
<td>Malay: Saya ada seorang adik.</td>
<td></td>
</tr>
<tr>
<td>English: I <em>goes</em> to Melaka.</td>
<td>I go to Melaka.</td>
</tr>
<tr>
<td>Malay: Saya pergi ke Melaka.</td>
<td></td>
</tr>
<tr>
<td>English: He <em>like</em> to play football.</td>
<td>He likes to play football.</td>
</tr>
<tr>
<td>Malay: Dia suka bermain bola.</td>
<td></td>
</tr>
<tr>
<td>English: My country <em>are</em> so beautiful.</td>
<td>My country is so beautiful.</td>
</tr>
<tr>
<td>Malay: Negara saya sangat cantik.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.38: Results of Statistical Significance Tests for the $\text{sva}(X)$ error category

<table>
<thead>
<tr>
<th>Error Category</th>
<th>Statistical Tests</th>
<th>Statistically significant?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{sva(verb)}$</td>
<td>$W=469.5, p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>$\text{sva(have)}$</td>
<td>$W=361.5, p&gt;0.05$</td>
<td>No</td>
</tr>
<tr>
<td>$\text{sva(be)}$</td>
<td>$W=441, p&gt;0.05$</td>
<td>No</td>
</tr>
</tbody>
</table>

may be related to the transfer rule problem.

Now, only the fifth error category in Table 3.35 which is the $\text{sva}(X)$ error category is yet to be explained. Table 3.37 shows some examples of the students’ sentence which have the subject verb agreement error. The statistical test shows that there is no significant difference between both groups. I also performed the statistical test on each argument type; the results are depicted in Table 3.38. Again, the results show there is no statistically significant difference for $\text{sva(verb)}$, $\text{sva(have)}$, and $\text{sva(be)}$ between Group 1 and Group 2. What can I conclude from these results of this longitudinal comparison study? I will discuss this in the next section.
3.8.2.3 Discussion

The significance test results show that in general, Group 2 (higher level) performs better in committing less errors than Group 1 (lower level). Grammar constructions that Group 2 has improved are the use of the copula *be*, the auxiliary *will* and the right use of tense words. Generally, this information will tell language teachers that their students are improving in acquisition of those constructions. Nevertheless, Group 2 has not progressed in certain grammar rules, namely the use of determiners and the subject-verb agreement (sva) rule. Statistical test results show no significant difference for these two grammar rules between Group 1 and Group 2. These prove that Group 2’s students are still not improving in those constructions. Although the determiners and the sva rule do not exist in L1 grammar, transfer issues can not be used as the only reason. This is because the copula *be* and the tense form also do not exist in L1 grammar but Group 2 has shown the improvement. Mohideen (1996) highlights some contributory factors to students’ errors. Besides L1 interference, I will highlight two factors: insufficient exposure to proper English usage, and insufficient time spent for grammar teaching and learning.

The first factor relates to the social context of English usage. This relates to “*inadequate exposure to the target language*” factor as mentioned in Mohideen (1996). English is taught as a foreign language in Malaysia and the participating schools in my study are of type sub-urban schools. Unlike urban area or big cities, people who live in sub-urban areas are not widely exposed to English. The students’ environment does not promote the use of English, the students are embarrassed to use English. For example, amongst the students, English is only used in school, especially during English class. Most students speak their own mother-tongue at home. Even whilst communicating among different races, the students tend to speak Malay. This lack of practice in conversing in English contributes to the students’ inability to master certain aspects of English grammar. When the students have a limited exposure to English, it is possible for them to make errors on those rules.

English teachers should advise their students to listen to good English from whatever available source i.e. radio, television, native speakers and good local speakers of
the language. However, the students may use sources which do not use good English. Mohideen (1996) claims “the possibility that errors could result from the inevitable exposure to non-native speakers of English teaching subjects such as history, geography and science in schools”. For example, the students may listen to how their teachers speak English especially Mathematics and Science teachers.\(^6\) A research conducted by Juriah Long and colleagues reveals that 60% of Mathematic and Science teachers are not proficient in English while 45% of the teachers are comfortable teaching in the language (The Star, 2006).

The second factor relates to how much time spent for teaching and learning grammar. The communicative approach has been introduced in language teaching and learning in Malaysian schools since 1983. The approach strongly encourages a communication between teachers and students by using real-life lesson topics in teaching materials (Galloway, 1993). In this concept, rather than learn about language grammar rules, the learners learn how to use the language in a certain situation. However, basic English constructions have been introduced to students since they start schooling in the primary level. As the students move to higher levels in primary and secondary schools, more constructions are introduced. The teaching syllabus can be found in KBSR and KBSM\(^7\) curriculum (Kementerian Pendidikan Malaysia, 2001, 2000). Both syllabi are organised around topics of themes which means the teaching of grammar is done in context and is not straightforward. On the other hand, examination systems focus on testing students’ grammatical ability. This becomes a conflict between the syllabus and examination and leaving the teachers in a dilemma over what to teach. If they are trying to keeping up with the given syllabus, the language teachers may spend little or no time teaching grammar rules. If that happens, revision of some grammar constructions that already learnt by the students won’t take place. Neglecting or less revision of learned grammar rules may lead the students to forget about them.

Teachers’ concern about their students’ grammar ability are discussed in Asraf’s study. The study investigates the teachers’ attitudes toward the KBSM syllabus. A

---

\(^6\)Since 2003, Mathematics and Science subjects are taught in English language which implementing a new policy in the national education system announced by the Malaysian government.

\(^7\)KBSR curriculum is for primary school and KBSM curriculum is for secondary school.
group of 419 English teachers responded to a questionnaire regarding several aspects of the KBSM syllabus. One of the questions asked is about teaching grammar in context. In response, more than 90% of the teachers prefer to spend more time on grammar teaching. For instance, one of the teachers commented, “I feel that grammar should not be integrated while teaching the language because by doing so it may confuse students. Grammar must be taught separately so that students are confident to use the language.” Further discussion about the KBSM syllabus and teachers’ views and comments on the syllabus can be found in Asraf (1996); Pillay and North (1997).

3.9 Summary

This chapter described an empirical study of learners’ error responses to some typical questions or dialogues in daily conversation. Responses were collected from pupils in three different secondary schools in Malaysia. Stages I went through in the study are described: the creation of an error classification scheme, the measuring of inter-coder agreement in error annotation tasks, and the analysis of corpus data.

The error classification scheme was used as a guideline to annotate students’ errors. There are six error categories which were represented in predicate form. Each predicate has either no, one or two arguments.

The $\alpha$ and $\kappa$ reliability tests were performed to calculate the inter-coder agreement. The reliability measurement was performed on two levels: predicate and argument levels. The results have shown good inter-coder reliability which allowed subsequent error annotation to be done for the remaining of corpus data.

The data analysis was done to find answers for two questions: “What are the most common errors made by the students?” and “How does performance of the students change across form levels?” The two most frequent errors made are missing a determiner or a copula and incorrect use of tense. These errors could be related to the non-existence of the rules in the students’ L1. In response to the second question, statistical tests were performed. The results proved that there is a significance different between two groups which means students performed better as they went through language learning process over time. However, there are some grammar rules which
both higher level and lower level students are still making errors. It shows that high level students are still having difficulties acquiring the use of sva rules and determiners.

L1 interference, insufficient exposure to proper English usage, and insufficient time spent for grammar learning are some contributory factors to students’ errors. However the analysis results are insufficient to help me in implementing my own error correction method. Using the existing learner corpus, I decide to develop a statistical model of error correction instead. In the following chapter, I will discuss the design of the proposed new error correction model.
Chapter 4

A Statistical Model of Error Correction

Remember my goal is to build an error correction system. In this chapter, I develop a statistical error correction model. The model will be trained (and tested) using the data gathered on the empirical study reported in Chapter 3.

My original goal was to develop specialised error correction routines to tackle the most common errors. But when analysing the error data, there is no obvious specific word error category among the frequent ones. Since the error data consists of many types of error, it is hard for me to only focus on a specific type of word error. This has led me to search for an alternative to the original plan.

The corpus of learner data that I gathered during the empirical study is a very rich source of information. It is not only the learners’ personal data, but also has other valuable contents, such as the learners’ responses both grammatical and ungrammatical, the proposed corrections for each ungrammatical sentence, and the related errors. With a database of erroneous sentences and their proposed corrections, a “surface-based” model of error correction is able to be developed. The model describes how an erroneous sentence can be corrected based on the most common surface transformations. For instance, an erroneous sentence “They goes to school by bus” is commonly corrected as “They go to school by bus”. We can use data of this sort to develop a statistical model of error correction. The statistical model will be based on
language modelling (LM) techniques (Jurafsky and Martin, 2009). As discussed in §2.7.2, language modelling techniques are statistical methods which forecast how likely it is that a word occurs in a particular surface context. The estimation is based on the number of occurrences of the word, in the sentence, in a corpus or database. The technique inspires me to build a model of perturbation likelihood based on a similar language modelling methodology. Thus while a regular language model estimates the likelihood of a particular word in a given context, I develop a model of the likelihood of a particular correction in a given context. This is a new idea in language modelling and error correction. So I need to introduce it gradually.

The following sections will explain my work in modelling the statistical error correction. I will first explain a current error correction module in the Kaitito system in §4.1. §4.2 will describe the background of language modelling. Then, my proposed error correction model will be further explained in §4.3, §4.4 and §4.5. Lastly before summarising the chapter, §4.6 and §4.7 will describe respectively an evaluation of the error correction model and results of the evaluation.

4.1 Error Correction Module in The Kaitito System

The Kaitito system is a multi-language dialogue system which is designed to assist language learners to learn a new language. Currently the system supports two languages: English and Māori. When the language learner enters an input sentence, the system responds to the sentence. This section describes briefly how the response takes place. The process of responding involves two phases: a perturbation phase, and an interpretation and disambiguation phase. Figure 4.1 depicts the flow of the perturbation module and the interpretation and disambiguation module in the Kaitito system.

First and foremost, I will describe the interpretation and disambiguation phase in §4.1.1, then the perturbation module in §4.1.2. This is because my focus of work is on the perturbation module.
4.1.1 Interpretation and Disambiguation Module

As shown in Figure 4.1, when a language learner enters a sentence to Kaitito, it first undergoes a perturbation process. Further explanation of the process will be given in §4.1.2.

An output sentence from the perturbation process is then passed to the interpretation and disambiguation module. The sentence is interpreted in three stages: the syntactic level, the semantic level and the dialogue act level, which takes place in a pipeline form. At the syntactic process, the sentence is parsed using the LKB system (Copestake and Flickinger, 2000). The parsing results are a list of syntactic parse trees of the sentence. The list is then passed to the semantic level. A semantic representation is derived from each parse tree at the end of the semantic process. The representation consists of a list of referring expressions which may have presuppositions that need to be resolved. The resolving of presupposition can be finding referents for the anaphora in the input sentence. For instance, an input sentence is A cat chases the mouse. The word the is an anaphora and must have been informed in previous input sentences.
Finally the semantic expressions are forwarded to the dialogue act level. At this stage, a set of updates is created for each semantic expression. From the updates, the input sentence is analysed to determine whether it is a question, an answer to a question, an assertion or a clarification question. Once the type is identified, Kaitito decides how to respond to the learner’s sentence. Along the interpretation process, the sentence is disambiguated at every level.

A sentence disambiguation is a process of making decisions on which of the outputs of every level is the most intended meaning of the sentence. Information derived from each level is taken into account before making a decision. At the syntactic level, a parse tree with highest probabilistic value is considered. At the semantic level, a referring expression with fewest presuppositions is selected. At the dialogue level, an update which discharges items from a dialogue stack is preferred. If there is only one interpretation, it is selected and passed to the next level. However, if more than one interpretation is generated, clarification questions are posed. See Vlugter, van der Ham, and Knott (2006) and Lurcock, Vlugter, and Knott (2004) for more details on how the interpretation and disambiguation module works.

4.1.2 Perturbation Module

Before an interpretation and disambiguation module is executed, an input sentence first goes through a perturbation process. The perturbation is an operation to transform or perturb the sentence into a list of different sentences. For example a sentence

(21) *We studying together.*

can be perturbed in some possible ways such as

(22) a. *We are studying together.*

b. *Studying together.*

c. *We study together.*

d. *We together studying.*

The idea of the perturbation module was suggested by Lurcock (2005). Later van der Ham (2005) developed the module. van der Ham developed two levels of perturba-
tion: **character-level** perturbations and **word-level** perturbations. Character-level perturbations are intended to correct typos and spelling errors, while the word-level perturbations is intended to perturb grammatical errors. Both levels perform four types of **perturbation actions**. The actions are *inserting* an additional character/word, *deleting* a character/word, *substituting* a character/word with another character/word, and *transposing* two adjacent characters/words. The basic idea of applying these actions was inspired by Kukich (1992). Kukich identifies that 80% of misspelled words require one of four types of actions to correct the spelling. Example (22a), (22b), (22c), and (22d) respectively show the result when the action of insertion, deletion, substitution, and transposition is applied on the sentence (21).

The output of the perturbation operation is a **perturbation list**. The list consists of several hypotheses in which each hypothesis comprises a perturbed sentence, a perturbation action and a **perturbation score**. The perturbed sentence is a possible correction of the input sentence. Since perturbations are made without reference to the system’s grammar, the perturbed sentence can be grammatical or ungrammatical. The perturbation score is a **conditional probability** value that fall within the range of 0.0 to 1.0. The probability value indicates how likely it is that the perturbed sentence is the preferred correction given the input sentence. A higher perturbation score means the correction is more likely. In mathematical notation, the probability is represented as $P(\text{perturbed sentence} \mid \text{input sentence})$. The symbol is interpreted as a probability of how likely a perturbed sentence is generated based on an input sentence. Further details about how the perturbation scores are calculated will be given in §4.3.

To clarify, consider an example. In this example and for the rest of this chapter, I only focus on word-level perturbations. This is because my work focuses on grammatical corrections. Suppose the input sentence is the sentence (21), *We studying together* and the sentence can be perturbed to a list of possible word-level perturbations as provided in (22), on page 128.

A key question for this approach is: "How are the perturbations generated?". There are too many possible perturbations to try them all, of course. Realistically, the perturbation operation can only be performed by making reference to a model of learners’ errors. To built such a model, we need a **corpus of perturbations**. The corpus
consists of a set of erroneous sentences and their corrections. I have a corpus of this kind, as described in the previous chapter.

Using a model of learners’ errors developed from the corpus, we can generate a list of candidate perturbations, and assign each perturbation a probability. Looking ahead to the end of this process, here is a list of candidate perturbations, ranked by probability.

(23) a. \( P(\text{We are studying together} \mid \text{We studying together}) = 0.4 \)
    
b. \( P(\text{We study together} \mid \text{We studying together}) = 0.25 \)
    
c. \( P(\text{Studying together} \mid \text{We studying together}) = 0.2 \)
    
d. \( P(\text{We together study} \mid \text{We studying together}) = 0.004 \)

The higher score shows how likely it is that the perturbed sentence will become the most preferred correction based on the corpus. In this example, the highest score is 0.4. We may suggest that \textit{We like studying together} would be the most preferred correction for the sentence \textit{We studying together}. This is only an example because the perturbation lists may end up with many kinds of perturbation. Since perturbations are made without reference to the system’s grammar, the perturbed sentence can be grammatical or ungrammatical. We may just pick the three highest scores as the suggested correction, and ignore the rest.

van der Ham (2005) developed a basic model to calculate the perturbation score. However, van der Ham only manages to evaluate his model with a very small set of data. This raises issues of validity. This leads me to restructure the model and propose a new perturbation model. The model is developed based on language modelling techniques. The techniques will be described further in §4.2, and I will describe my own model in §4.3.

4.2 Language Modelling

Statistically modelling a language is a process which represents probabilistic information about word sequences in a particular domain of interest (Jurafsky and Martin, 2009). The probabilistic information takes the form of probability values that denote
how likely word sequences are in that domain. Suppose a sentence starts with the following words:

(24) You are so . . .

Most likely the following word in (24) would be an adjective such as beautiful or horrible. It is unlikely it would be a preposition such as with or a noun such as book. Between the word beautiful and horrible, how could we predict which one is the most likely word used? The prediction can be done by computing how frequently beautiful and horrible occur in this context in a naturally occurring corpus. A corpus is a collection of actual text or speech (normally a very large one these days). A context is a sequence of words adjacent to or surrounding the word we want to predict. Contexts have to be relatively short sequences because it is very unlikely that a particular long sequence will be found in a corpus, especially if it is very big. For instance, say we want to predict the last word in the sentence

(25) My father, my grandfather, and my uncle, Saiful went for a holiday together last Christmas holiday and they . . .

It is unlikely we will find this exact sentence in a training corpus. So contexts have to be a short sequence of adjacent words. This approximation is known as an n-gram model. An n-gram model is a sequence of n words of the language model. A 1-gram or unigram consists of a word only. A 2-gram or bigram consists of 2 words, 3-gram or trigram has 3 words, and so on. An example in (26)\(^1\) shows a very small context from the Google trigram corpus. Note that the number provided at the end of each trigram indicates how many times each occurs in the trigram corpus.

(26) ceramics collectables collectibles 55
    ceramics collectables fine 130
    ceramics collected by 52
    ceramics collectible pottery 50
    ceramics collectibles cooking 45

\(^1\)The example is referred from http://en.wikipedia.org/wiki/N-gram. The date accessed was on 4th July 2011.
4.2.1 Maximum Likelihood Estimate

The maximum likelihood estimate (MLE) is the simplest technique for calculating the probability of a word in a given context. The higher probability value, the more likely it is that the word will appear. Suppose we want to know the MLE probability of a single word, \( w_1 \) occurs in a training corpus. The basic formula of MLE probability for the word \( w_i \) found in a training corpus, \( P_{\text{mle}}(w_i) \) is defined as:

\[
P_{\text{mle}}(w_i) = \frac{\text{cnt}(w_i)}{N}
\]

(4.1)

where

- \( \text{cnt}(w_i) \) is how many times \( w_i \) is found in the corpus. We can also refer to it as the count of \( w_i \).
- \( N \) is the total numbers of word tokens in the corpus.

\( N \) refers to all words in the training corpus including recurring ones and excluding punctuation. For instance, in the sentence (25), \( N \) is 18.

In calculating probabilities of longer n-grams, \( P(w_1w_2 \ldots w_n) \), approximations need to be made. This is typically done by decomposing the joint probability into a product of conditional probabilities. The MLE for a conditional probability is as follows:

\[
P_{\text{mle}}(w_n|w_1w_2 \ldots w_{n-1}) = \frac{\text{cnt}(w_1w_2 \ldots w_{n-1}w_n)}{\text{cnt}(w_1w_2 \ldots w_{n-1})}
\]

(4.2)

As an example, we want to know the MLE probability of \( \text{You are beautiful} \) in a training corpus, \( P_{\text{mle}}(\text{beautiful}|\text{You are}) \). The probability is read as a probability of beautiful with a condition that the word is after \( \text{You are} \). It is also known as a conditional probability. The probability is estimated by computing the count of the trigram \( \text{You are beautiful} \) in a training corpus, and dividing it by how many times the bigram \( \text{You are} \) occurs in the corpus. Below are the mathematical symbols that will be used.

- \( P_{\text{mle}}(\text{beautiful}|\text{You are}) \) represents the conditional probability that the word beautiful appears in the context \( \text{You are} \).
- \( \text{cnt}(\text{You are beautiful}) \) indicates the number of times \( \text{You are beautiful} \) occurs in the training corpus.
• \( \text{cnt}(\text{You are}) \) represents the number of times \( \text{You are} \) in the training corpus.

Therefore,

\[
P(\text{beautiful}|\text{You are}) = \frac{\text{cnt}(\text{You are beautiful})}{\text{cnt}(\text{You are})} \tag{4.3}
\]

Now I will give some examples of MLE calculation. Suppose a very small training corpus with 4 sentences is shown in (27) below.

(27)  <sos> My parents are kind <eos>

<sos> I am happy <eos>

<sos> The happy girl smiled at me <eos>

<sos> I am not happy <eos>

The symbol <sos> represents the start of sentence and <eos> represents the end of sentence. Suppose we want to calculate a probability of happy given am, so the MLE formula is

\[P_{\text{mle}}(\text{happy}|\text{am}) = \frac{\text{cnt}(\text{am happy})}{\text{cnt}(\text{am})} \tag{4.4}\]

From the training corpus (27), the \( \text{cnt}(\text{am happy}) \) is 1 and \( \text{cnt}(\text{am}) \) is 2, therefore

\[P_{\text{mle}}(\text{happy}|\text{am}) = \frac{1}{2} = 0.5 \tag{4.5}\]

The following are other examples of calculating probabilities for some bigram models.

(28)  a. \(P_{\text{mle}}(I |<\text{sos}> )= \frac{\text{cnt}(<\text{sos}>I)}{\text{cnt}(<\text{sos}>)} = \frac{2}{4} = 0.5\)

b. \(P_{\text{mle}}(<\text{eos}|\text{kind}) = \frac{\text{cnt}(\text{kind}<\text{eos}>)}{\text{cnt}(\text{kind})} = \frac{1}{1} = 1.0\)

c. \(P_{\text{mle}}(\text{girl}|\text{happy}) = \frac{\text{cnt}(\text{happy girl})}{\text{cnt}(\text{happy})} = \frac{1}{3} = 0.33\)

The above examples indicate when all the given bigram words appear in the training corpus. What will happen when a word sequence is unseen in the training corpus, i.e. \(\text{happy woman}\)? Calculating the MLE probability for unseen words yields a zero value. When a word or a sequence of words does not occur in a corpus, it does not mean there
is a zero probability of it occurring. The event may not occur in the corpus but the corpus is only finite. The number of possible word sequences in a language is so huge that a corpus can not manage to contain all of them - even for quite short sequences. This situation is known as a zero-probability problem or a sparse data problem. To prevent such a problem from happening, a smoothing technique is applied.

### 4.2.2 Smoothing

A smoothing technique is a technique which addresses the problem of sparse data. Let me define term an event as a word sequence to be found in a training corpus. When an event occurs many times in the corpus, its probability should be high. On the other hand for an other event which is rarely seen or unseen in the corpus, its probability may be very low or zero.

In the smoothing process, some of the probability mass for events is redistributed to rare events and unseen events. It is done by decreasing the probability of seen events and reassigning it to low and zero count events. After the smoothing process, all counts have a new value which is known as the adjusted counts. Figure 4.2 describes the situation happening to the probability for seen, low count and unseen events before and after the smoothing technique is applied. The following subsection will explain two smoothing techniques of how the zero problem is encountered differently. The techniques I will describe are Add-One and Witten-Bell.
4.2.2.1 Add-One Smoothing

Add-One smoothing is a simple smoothing technique. To avoid the zero event problem, the count of each possible event is adjusted by adding one. We know that the MLE formula for an event $w_i$ is defined as

$$P_{\text{mle}}(w_i) = \frac{\text{cnt}(w_i)}{N} \quad (4.6)$$

To apply the Add-One concept to Equation 4.6, $\text{cnt}(w_i)$ is incremented by one. Since the numerator of the MLE fraction is incremented, its denominator, $N$ has to be adjusted also. The denominator is added to by the total numbers of word types, $V$ in a training corpus. Therefore the Add-One formula is formulated as

$$P_{\text{add-one}}(w_i) = \frac{\text{cnt}(w_i) + 1}{N + V} \quad (4.7)$$

$V$ refers to all words in a corpus excluding recurring ones. Meanwhile the word tokens, $N$ refers to all words including the recurring ones. For instance, in a sentence *The cat and the dog were eating the cake*, $N$ is 9 and $V$ is 7.

An adjusted count is generated as the result of smoothing techniques. Sometimes it is useful to compare an original count with its respective adjusted count in order to see how it varies the smoothing technique by change the count. The adjusted count, $\text{cnt}^*$ is computed by multiplying Equation 4.7 by $N$. Therefore the formula of $\text{cnt}^*$ is

$$\text{cnt}^*_{\text{add-one}}(w_i) = (\text{cnt}(w_i) + 1) \times \frac{N}{N + V} \quad (4.8)$$

In Equation 4.8, to get a probability value, we just normalise the formula with $N$. As a result, we again get the equation as depicted in Equation 4.9. Further information regarding $\text{cnt}^*$ will be further explained in §4.2.3.

A generalisation of Add-One formula in an $n$-gram model is below

$$P_{\text{add-one}}(w_n|w_1w_2...w_{n-1}) = \frac{\text{cnt}(w_1w_2...w_{n-1}w_n) + 1}{\text{cnt}(w_1w_2...w_{n-1}) + V} \quad (4.9)$$

Based on Example (28) on page 133 and the corpus (27) on page 133, by applying Equation 4.9, new probability values are shown in (29) below. The comparison between each probability value yielded from MLE and Add-One techniques is outlined in Table 4.1.
Table 4.1: Comparison of each probability value between MLE and Add-One.

<table>
<thead>
<tr>
<th></th>
<th>MLE</th>
<th>Add-One</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(I</td>
<td>&lt;sos&gt;)$</td>
<td>0.5</td>
</tr>
<tr>
<td>$P(&lt;eos&gt;</td>
<td>kind)$</td>
<td>1.0</td>
</tr>
<tr>
<td>$P(girl</td>
<td>happy)$</td>
<td>0.33</td>
</tr>
<tr>
<td>$P(man</td>
<td>happy)$</td>
<td>0</td>
</tr>
</tbody>
</table>

(29)  
\[ P_{\text{add-one}}(I|<sos>) = \frac{\text{cnt}(<sos>\ I)+1}{\text{cnt}(<sos>)+V} = \frac{2+1}{4+15} = \frac{3}{19} = 0.16 \]

b. \[ P_{\text{add-one}}(<eos>|kind) = \frac{\text{cnt}(kind <eos>)+1}{\text{cnt}(kind)+V} = \frac{1+1}{1+15} = \frac{2}{16} = 0.13 \]

c. \[ P_{\text{add-one}}(girl|happy) = \frac{\text{cnt}(happy girl)+1}{\text{cnt}(happy)+V} = \frac{1+1}{3+15} = \frac{2}{18} = 0.11 \]

d. \[ P_{\text{add-one}}(man|happy) = \frac{\text{cnt}(happy man)+1}{\text{cnt}(happy)+V} = \frac{0+1}{3+15} = \frac{1}{18} = 0.06 \]

As we can see in Table 4.1, the difference between MLE and Add-One probability values is quite big especially for higher values. For example, the $P_{\text{add-one}}(I|<sos>)$ decreases more than half as compared to $P_{\text{mle}}(I|<sos>)$, which is from 0.5 to 0.16. Another drastic reduction is $P_{\text{mle}}(<eos>|kind)$ which is from 1.0 to 0.13 in Add-One! Although zero probability problems won’t occur, adding one to each count causes too much probability mass to be passed to zero and low count $n$-grams. That is why there is a big gap of probability values between higher counts $n$-gram in MLE and Add-One. Therefore, other smoothing techniques such as Witten-Bell smoothing are introduced.

4.2.2.2 Witten-Bell Smoothing

The Witten-Bell (WB) technique is another technique to solve the problem of zero probability. Here, the count of unseen events is estimated by calculating the occurrence of events which we are seeing for the first time (Witten and Bell, 1991). We can assume

\[ \text{Method C in Witten-Bell smoothing is used here.} \]
that a training corpus is a number of events. \( T \) is a count of the number of times a new event type is seen in the corpus - i.e. a count of the number of event types in the corpus, and \( N \) is how many event tokens occur in the corpus. Let me give an example. In a corpus consisting of seven sentences:

\[
(30) \quad \begin{align*}
&a. \ You \ are \ kind. \\
&b. \ I \ am \ happy. \\
&c. \ I \ am \ happy. \\
d. \ He \ is \ happy. \\
e. \ I \ am \ unhappy. \\
f. \ You \ are \ kind. \\
g. \ I \ am \ happy. 
\end{align*}
\]

Each sentence in the corpus (30) is counted as an event token, so there are 7 event tokens, \( N \). Then each different sentence is considered as an event type. Since there are 4 different sentences which are \( You \ are \ kind, \ I \ am \ happy, \ He \ is \ happy, \) and \( I \ am \ unhappy, \) so the event types, \( T \) is 4.

Now let’s focus on how the WB formula is defined. The WB probability for an event, \( e_i \) is calculated by normalising the count of \( e_i \) with the sum of all possible event types, \( T \) and event tokens. Hence the equation is

\[
P_{wb}(e_i) = \frac{\text{cnt}(e_i)}{T + N}, \text{ if } \text{cnt}(e_i) > 0 \quad (4.10)
\]

The normalisation by \((T + N)\) is applied in Equation 4.10 because \( T \) and \( N \) are treated as two different events. What may happen if \( \text{cnt}(e_i) = 0 \)? Surely we know that if that case occurs, it is considered as a zero probability in MLE. In WB smoothing, the zero count event is estimated from the count of event types, \( T \). We can estimate a total probability mass of all zero counts by dividing \( T \) by the sum of \( T \) and \( N \). Hence the equation of the total probability mass of zero-counts is formulated as:

\[
\sum_{e_i: \text{cnt}(e_i) = 0} P_{wb}(e_i) = \frac{T}{T + N} \quad (4.11)
\]

Then how do we specify a probability of each zero count event? This can be done by dividing the probability mass by the total number of zero count events. We define \( Z \) to
be the total number of zero count events, therefore the probability of each zero count event is
\[ P_{wb}(e_i) = \frac{T}{Z(T+N)}, \text{ if } \text{cnt}(e_i) = 0 \] (4.12)

For some given \( w_1 \ldots w_{n-1} \), the total space of events \((e_i)\) of the form \( w_1 \ldots w_{n-1}w_n \) will be of size \( V \), as \( w_n \in V \). \( Z \) is therefore this value less the events from that space that have already been seen, i.e. \( T \). A summary of Equation 4.10 and 4.12 is depicted below,

\[ P_{wb}(e_i) = \begin{cases} \frac{\text{cnt}(e_i)}{T+N}, & \text{if } \text{cnt}(e_i) > 0 \\ \frac{T}{Z(T+N)}, & \text{if } \text{cnt}(e_i) = 0 \end{cases} \] (4.13)

Equation 4.13 is applied to unigram models. How do we generalise it for \( n \)-gram models, for example a bigram model, a trigram model and so forth? In order to calculate WB probability for \( n \)-gram (here \( n \) is greater than one), information about its \((n-1)\)-gram is required. When calculating a probability of non-zero count \( n \)-grams, information of its \((n-1)\)-gram types, \( T \) and tokens, \( N \) is used. Similarly, a probability of zero count \( n \)-gram is calculated based on \( T \) and \( N \) as well. Equation 4.14 below represents the WB formula for \( n \)-gram models.

\[ P_{wb}(w_n|w_1 \ldots w_{n-1}) = \begin{cases} \frac{\text{cnt}(w_1 \ldots w_{n-1}w_n)}{T(w_1 \ldots w_{n-1})+N(w_1 \ldots w_{n-1})}, & \text{if } \text{cnt}(w_1 \ldots w_{n-1}w_n) > 0 \\ \frac{T(w_1 \ldots w_{n-1})}{Z(w_1 \ldots w_{n-1})(T(w_1 \ldots w_{n-1})+N(w_1 \ldots w_{n-1}))}, & \text{if } \text{cnt}(w_1 \ldots w_{n-1}w_n) = 0 \end{cases} \] (4.14)

Let’s start with a simple example. Suppose in corpus \((30)\), we want to calculate a probability of a trigram \( I \ am \ happy \). An \((n-1)\)-gram for \( I \ am \ happy \) is a bigram \( I \ am \). There are three occurrences of \( I \ am \) in the corpus, so \( N(I \ am) = 4 \). Yet, \( N(I \ am) \) can also be represented as \( \text{cnt}(I \ am) \). \( T(I \ am) \) means how many different types of \( n \)-gram start with \( I \ am \). Here, there are two types that starts with \( I \ am \) which are \( I \ am \ happy \) and \( I \ am \ not \ happy \). As such \( T(I \ am) = 2 \). If \( \text{cnt}(I \ am \ happy) > 0 \), then we can just apply those values to an appropriate equation in 4.14.

On the other hand, what if \( \text{cnt}(I \ am \ happy) = 0? \) Referring to Equation 4.14 for a case of zero counts, there is a variable called \( Z \). Hence, \( Z(I \ am) \) represents a total
number of zero count (unseen) \( n \)-grams, which means, here, how many unseen \( n \)-grams start with \( I \ am \). This variable \( Z(I \ am) \) is calculated by subtracting \( T(I \ am) \) from the vocabulary size of the corpus, \( V \). In a mathematical formula, \( Z(I \ am) = V - T(I \ am) \).

The next paragraph provides an example of WB calculation I have previously performed in MLE and Add-One sections.

These examples refer to the corpus (27) on page 133 and Example (28) on page 133. Details of computation and results are outlined in (31). A comparison among each probability value computed from MLE, Add-One and WB techniques is depicted in Table 4.2.

\[
\begin{align*}
\text{Table 4.2: Comparison of each probability value among MLE, Add-One and WB.} \\
\begin{array}{|c|c|c|c|}
\hline
 & \text{MLE} & \text{Add-One} & \text{WB} \\
\hline P(I|<\text{sos}>)) & 0.5 & 0.16 & 0.29 \\
P(<\text{eos}>|\text{kind}) & 1.0 & 0.13 & 0.5 \\
P(\text{girl}|\text{happy}) & 0.33 & 0.11 & 0.2 \\
P(\text{man}|\text{happy}) & 0 & 0.06 & 0.03 \\
\hline
\end{array}
\end{align*}
\]

As depicted in Table 4.2, the difference between WB and MLE probability values is not so great. For instance, \( P_{\text{wb}}(I|<\text{sos}>)) \) reduces approximately a half from \( P_{\text{mle}}(I|<\text{sos}>)) \). Similarly in \( P_{\text{wb}}(<\text{eos}>|\text{kind}) \), the probability drops half from \( P_{\text{mle}}(<\text{eos}>|\text{kind}) \). Therefore, we can say here that the WB technique yields the probability values which are more reasonable than the Add-One technique.
Jurafsky and Martin (2009) view a discounting technique as an alternative to a smoothing algorithm. This technique is applied in order to get a probability mass by decreasing non-zero counts $n$-grams and assigns the probability mass to zero counts $n$-grams. A discount value, $\text{discnt}$ is generated from the discounting technique and is defined as

$$\text{discnt}(w_i) = \frac{\text{cnt}^*(w_i)}{\text{cnt}(w_i)}$$

(4.15)

If we would like to see how far an original count is adjusted to a new count, an adjusted count for WB, $\text{cnt}_{wb}^*(w_i)$ is computed by multiplying each sub-equation in Equation 4.13 by $N$,

$$\text{cnt}_{wb}^*(w_i) = \begin{cases} 
\text{cnt}(w_i) \times \frac{N}{T+1}, & \text{if } \text{cnt}(w_i) > 0 \\
\frac{T}{Z} \times \frac{N}{T+1}, & \text{if } \text{cnt}(w_i) = 0
\end{cases}$$

(4.16)

In the following section, I’ll describe a technique that applied a discounting technique and also uses information of $(n-1)$-grams if there is no occurrence of its $n$-grams. The technique is called Backoff.

### 4.2.3 Backoff

A backoff technique is another technique to solve zero probability problems. The backoff technique was introduced by Katz (1987). The technique suggests that if there is no occurrence of a particular $n$-gram, we may back off to lower order $n$-grams (i.e. $n$-$1$-grams), which are likely to have higher counts. If these too have zero counts, we back off again to $(n-2)$-grams. The back off process continues until we find some counts of the lower order models. To clarify with an example, suppose if $P(\text{happy} \mid I\ am) = 0$, we can estimate the probability by looking at $P(\text{happy} \mid am)$. Otherwise, we back off to the unigram and calculate $P(\text{happy})$.

The backoff algorithm in a trigram model is defined in Equation (4.17) below
P\text{\textsubscript{katz}}(w_3|w_1 w_2) = \begin{cases} 
\text{P}^\ast(w_3|w_1 w_2), & \text{Case 1: if cnt}(w_1 w_2 w_3) > 0 \\
\alpha(w_1 w_2) \text{P}^\ast(w_3|w_2), & \text{Case 2: else if cnt}(w_1 w_2 w_3) = 0 \\
\alpha(w_1) \text{P}^\ast(w_3), & \text{Case 3: otherwise.} 
\end{cases} 
(4.17)

Here I will describe the algorithm. As for Case 1, if there is an occurrence of a trigram \((w_1 w_2 w_3)\), its probability is derived from a discounted probability of the trigram. The discounted probability of the trigram is represented as \(\text{P}^\ast(w_3|w_1 w_2)\). As mentioned earlier in §4.2.2.2, a discounting technique is a process which assigns a probability mass to unseen \(n\)-grams. Therefore, \(\text{P}^\ast(w_3|w_1 w_2)\) is applied in order to set aside some probability mass to unseen trigrams. Assuming that all unseen \(n\)-grams have an equal probability value, the probability mass can be evenly distributed to all unseen \(n\)-grams. Meanwhile, the backoff technique has a better way to equally assign the probability mass, by depending on information from the lower order \(n\)-grams. In this case, the information from a bigram or a unigram is used to calculate a probability of an unseen trigram.

The discounted probability \(\text{P}^\ast\) for a trigram is formulated as:

\[
\text{P}^\ast(w_3|w_1 w_2) = \frac{\text{cnt}^\ast(w_1 w_2 w_3)}{\text{cnt}(w_1 w_2)} 
\]  
(4.18)

An adjusted count, \(\text{cnt}^\ast\), can be calculated by using WB formula as defined in Equation 4.16.\(^3\) The formula of MLE probability is

\[
\text{P}_{\text{mle}}(w_3|w_1 w_2) = \frac{\text{cnt}(w_1 w_2 w_3)}{\text{cnt}(w_1 w_2)} 
\]  
(4.19)

In comparison between \(\text{P}_{\text{mle}}(.)\) and \(\text{P}^\ast(.)\), the probability value of \(\text{P}^\ast(.)\) is always less than \(\text{P}_{\text{mle}}(.)\). This is because \(\text{P}^\ast(.)\) will leave some probability mass for unseen \(n\)-grams in which the mass can be passed to its lower order \(n\)-grams. A weight function, \(\alpha\), is applied to distribute the probability mass that will be further explained in the subsequent paragraphs.

\(^3\)In fact, Good-Turing smoothing (Good, 1953) is more commonly used than WB. I used WB because it was the method I was most familiar with.
In regard to Case 2, if a trigram is unseen \( \text{cnt}(w_1 w_2 w_3) = 0 \), then we back off to its bigram model, \((w_2 w_3)\). If the bigram is found, a probability of the trigram \( P_{\text{katz}}(w_3|w_2 w_1) \) is estimated by calculating a discounted probability of the bigram, \( P^*(w_3|w_2) \). Then the probability mass (for unseen trigrams) needs to be distributed to \( P^*(w_3|w_2) \). A weight function, \( \alpha(w_1 w_2) \) is applied to distribute the probability mass. As such, in order to pass the probability mass, we have to multiply \( P^*(w_3|w_2) \) by \( \alpha(w_1 w_2) \). If in turn we don’t observe any instances of \( (w_2 w_3) \), we back off to its unigram model, \((w_3)\).

Now let’s see how to calculate the weight function, \( \alpha \). In the \((w_1 w_2)\) context, suppose \( \beta(w_1 w_2) \) is a total discounted probability mass of unseen trigrams starting with \((w_1 w_2)\). Therefore, \( \beta(w_1 w_2) \) can be calculated by 1 minus the total discounted probability mass for all trigrams starting with \((w_1 w_2)\) context (Jurafsky and Martin, 2009) as defined below:

\[
\beta(w_1 w_2) = \sum_{w_3: \text{cnt}(w_1 w_2 w_3) = 0} P^*(w_3 | w_1 w_2) = 1 - \sum_{w_3: \text{cnt}(w_1 w_2 w_3) > 0} P^*(w_3 | w_1 w_2) \tag{4.20}
\]

As a result, Equation 4.20 yields the total discounted probability mass to be passed to all bigrams. However, to ensure that each individual bigram gets a fraction of the total discounted probability mass, we need to normalise the total discounted probability mass, \( \beta(w_1 w_2) \). Hence, to compute \( \alpha(w_1 w_2) \), we normalise \( \beta(w_1 w_2) \) by the total of all bigrams that start with unseen trigrams. As such the formula for \( \alpha(w_1 w_2) \) is defined as follows:

\[
\alpha(w_1 w_2) = \frac{\beta(w_1 w_2)}{\sum_{w_3: \text{cnt}(w_1 w_2 w_3) = 0} P_{\text{katz}}(w_3 | w_2)} = \frac{1 - \sum_{w_2: \text{cnt}(w_1 w_2 w_3) > 0} P^*(w_3 | w_1 w_2)}{1 - \sum_{w_3: \text{cnt}(w_1 w_2 w_3) > 0} P^*(w_3 | w_2)} \tag{4.21}
\]

If \( \text{cnt}(w_1 w_2) = 0 \), some assumptions are made as given below.

\[
P_{\text{katz}}(w_3 | w_1 w_2) = P_{\text{katz}}(w_3 | w_2) \text{ if } \text{cnt}(w_1 w_2) = 0 \tag{4.22}
\]
\( P^*(w_3 \mid w_1 w_2) = 0 \) if \( \text{cnt}(w_1 w_2) = 0 \) \hspace{1cm} (4.23)

and

\[ \beta(w_1 w_2) = 1 \] if \( \text{cnt}(w_1 w_2) = 0 \) \hspace{1cm} (4.24)

A general backoff formula for \( n \)-grams model which is adapted from Jurafsky and Martin (2009) is shown in Equation (4.25)-(4.30) below.

\[
P_{\text{katz}}(w_n \mid w_1 \ldots w_{n-1}) = \begin{cases} 
P^*(w_n \mid w_1 \ldots w_{n-1}), & \text{if } \text{cnt}(w_1 \ldots w_{n-1} w_n) > 0 \\
\alpha(w_1 \ldots w_{n-1}) P_{\text{katz}}(w_n \mid w_2 \ldots w_{n-1}), & \text{otherwise.}
\end{cases}
\] \hspace{1cm} (4.25)

where

\[
P^*(w_n \mid w_1 \ldots w_{n-1}) = \frac{\text{cnt}^*(w_1 \ldots w_{n-1} w_n)}{\text{cnt}(w_1 \ldots w_{n-1})} \] \hspace{1cm} (4.26)

and

\[
\alpha(w_1 \ldots w_{n-1}) = \frac{1 - \sum_{w_n: \text{cnt}(w_1 \ldots w_{n-1} w_n) > 0} P^*(w_n \mid w_1 \ldots w_{n-1})}{1 - \sum_{w_n: \text{cnt}(w_1 \ldots w_{n-1} w_n) > 0} P^*(w_n \mid w_2 \ldots w_{n-1})} \] \hspace{1cm} (4.27)

and when \( \text{cnt}(w_1 \ldots w_{n-1}) = 0 \), the following assumptions are made.

\[
P_{\text{katz}}(w_n \mid w_1 \ldots w_{n-1}) = P_{\text{katz}}(w_n \mid w_2 \ldots w_{n-1}) \] \hspace{1cm} (4.28)

and

\[
P^*(w_n \mid w_1 \ldots w_{n-1}) = 0 \] \hspace{1cm} (4.29)

and

\[
\beta(w_1 \ldots w_{n-1}) = 1 \] \hspace{1cm} (4.30)

In the following, I will give an example of how to calculate backoff probabilities. Here I will provide step by step calculation for non-zero and zero count \( n \)-grams. Since the example will be represented in a trigram model, Equation (4.17), (4.18), and (4.21)-(4.24) will be referred to. Here, I would like to calculate the backoff probability for a trigram \textit{I am happy} and \textit{They are kind} based on the corpus (30) on page 137. Firstly, I will start with the calculation of \textit{I am happy}. Since \textit{I am happy} is observed in the corpus,
according to Equation 4.17, Case 1 is fulfilled. The calculation of \( P_{\text{katz}}(\text{happy} | \text{I am}) \) is shown as follows:

\[
P_{\text{katz}}(\text{happy} | \text{I am}) = \frac{P^*(\text{happy} | \text{I am}) \cdot \text{cnt}^*(\text{I am happy})}{\text{cnt}(\text{I am})} = \frac{\text{cnt}(\text{I am happy}) \times \frac{N(\text{I am})}{T(\text{I am}) + N(\text{I am})}}{4} = 0.5
\]  

(4.31)

The adjusted count, \( \text{cnt}^*(\text{I am happy}) \) is computed by using the WB formula in Equation (4.16) on page 140. In order to refer back to how each of the above values is derived, see page 138.

Next, I will show how to calculate \( P_{\text{katz}}(\text{kind} | \text{They are}) \). Since there is no occurrence of \( \text{They are kind} \) in the corpus, we need to back off to the lower-order model. This means we need to find whether there is any occurrence of \( \text{are kind} \) in the corpus. Since the bigrams (\( \text{are kind} \)) are found in the corpus, Case 2 in Equation (4.17) is met. Below are the calculations:

\[
P_{\text{katz}}(\text{kind} | \text{They are}) = \alpha(\text{They are}) \times P^*(\text{kind} | \text{are})
\]

\[
= \frac{1 - \sum_{w_3: \text{cnt}(\text{They are } w_3) > 0} P^*(w_3 | \text{They are})}{1 - \sum_{w_3: \text{cnt}(\text{They are } w_3) > 0} P^*(w_3 | \text{are})} \times \frac{\text{cnt}^*(\text{are kind})}{\text{cnt}(\text{kind})}
\]

\[
= \frac{1}{1} \times \frac{\text{cnt}(\text{are kind}) \times \frac{N(\text{are kind})}{T(\text{are kind}) + N(\text{are kind})}}{2} = 0.667
\]  

(4.32)

The symbol \( w_3 \) represents any words which occur in the corpus. Therefore \( P^*(w_3 | \text{They are}) \) represents a discounted probability of any words that exist in the corpus, which is preceded by \( \text{They are} \). Based on the assumptions made in Equation 4.22 and 4.23, since \( \text{cnt}(\text{They are}) = 0 \), therefore \( P^*(w_3 | \text{They are}) = P^*(w_3 | \text{are}) = 0 \) in Equation 4.32.
4.3 The Proposed Model of Error Correction

In this section I will describe my proposed model of error correction. This model represents an algorithm for calculating perturbation scores. The model is developed based on the language modelling techniques just described. As mentioned in the previous section, language modelling techniques are statistical methods to forecast how likely it is that a word occurs in a particular surface context. The proposed model is a model of perturbation likelihood based on a similar language modelling methodology. While a regular language model estimates the likelihood of a particular word in a given context, the perturbation model estimates the likelihood of a particular perturbation of words in a given context. In the remaining chapters, the terms error correction and perturbation will be used interchangeably.

Given a sentence “You look happy and . . .”. In a language model, we want to estimate how likely it is that a word, say “cheerful” ends the sentence. Meanwhile, in my perturbation model, I want to estimate how likely it is that one string of words is perturbed/corrected to a different string of words. For instance, how likely it is that the sentence “I watches television” is corrected to “I watch television”.

My technique is reminiscent of Brockett, Dolan, and Gamon’s (2006) machine-translation-inspired approach to error correction (see §2.7.2.2), where a probability is calculated of one string of words being mapped onto another. I should say up front that I didn’t know about this approach when I devised my model. I admit that in lots of ways, the SMT approach is far better than the one I implement here. In particular, SMT promises to extend well to multiple errors, while my approach only handles single errors. However, what happen if a source language is unseen in a training data. My proposed model is able to propose corrections for an ill-formed sentence even though the sentence is not available in a learner corpus. For example, if the erroneous sentence “I watches television” is unseen in the corpus, this leads to the zero probability problem. In order to solve the problem, I implement a smoothing technique. If there is an occurrence of “I watches a movie” which is corrected to “I watch a movie”, so it is highly likely “I watches television” is corrected to “I watch television”. Ultimately, the corrections that I propose here should be similar to the corrections which I have
in my learner corpus - but they should also result in sentences which are statistically like normal English. So in some sense, I’m just focussing on one aspect of a machine-translation model.

The idea of using a noisy channel model for error correction has been around for quite a while such as Ringerger and Allen (1996), and yet it hasn’t revolutionised error correction. So obviously it is not a panacea for problems in error correction systems. It is worth noting that Brockett, Dolan, and Gamon use synthetic data, not data about real language learner errors. They manually generate synthetic errors with very simple templates a lot like my perturbations in reverse: e.g. converting the phrase ‘learn much knowledge’ to ‘learn many knowledge’. The more naturalistic corpus which I gather (described in Chapter 3) would be an ideal one to use with Brockett et al.’s translation based technique, and it’s something I’d like to try in future. Note, however, that there have been some recent experiments using MT-like sequence-mapping techniques on naturalistic datasets, for instance, Gamon (2011).

There are problems in a simple application of a MT approach to error correction: if the ‘source language’ sentences in the parallel corpus are only sentences with errors in them, then a MT-based approach will probably overgenerate error hypotheses. This is a problem that Gamon (2011) seems to have - see his discussion of ‘false positives’. Gamon claims that if his error detection system is incorporated with a component that suggests possible corrections and they can be ranked by a language model, then this can reduce the number of false positives by examining the language model ranked. Brockett et al. (2006) also mention that they need to have a parallel corpus which consists of erroneous sentences and their corresponding corrected version in order to provide native-like English sentences. In my system, I envisage using a statistical error-correcting technique alongside a wide-coverage symbolic grammar, and only invoking the error-correction system when the grammar fails to parse a student sentence. I also only suggest a correction to the student if the symbolic grammar can parse the proposed correction.
4.3.1 The Model

The model of perturbation probabilities will be based on a corpus of perturbations which consists of a list of pairs of sentences. The first sentence in each pair is an incorrect sentence given by an actual student, and the second sentence is the corrected sentence, as judged by a native speaker. This proposed model can only handle single word errors only. The generation of the corpus is described further in §4.4.

Here, I represent the difference between the two sentences as a set of one-word perturbations. I represent a one-word perturbation in a trigram model as $w_1 w_{\text{orig}} w_3$. I assume the second word of the trigram, $w_{\text{orig}}$ is the word to be perturbed. The word may be deleted or substituted with another word. Also the word may be transposed with its adjacent word, or a new word may be inserted in the sentence.

Given that the word to be perturbed is in the middle of the two context words, my perturbation model breaks the Markov assumption made in conventional $n$-gram probability models. This means that my implementation of Katz backoff is not guaranteed to result in a proper probability model. The model should more properly be thought of as a ‘scoring model’, which heuristically evaluates the goodness of different perturbations, and delivers scores for them which happen to range from 0 to 1. I will refer to the scores my model produces as “probability scores” rather than true probabilities. Suppose $w_{\text{orig}}$ is perturbed to $w_{\text{pert}}$, so the perturbation is represented as $(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3)$. I call the notation a perturbation rule. Therefore, I want to calculate how likely $(w_1 w_{\text{orig}} w_3)$ is perturbed to $(w_1 w_{\text{pert}} w_3)$. To do this, I assign a probability score $P(w_1 w_{\text{pert}} w_3 \mid w_1 w_{\text{orig}} w_3)$. Notationally, I write $P(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3)$ to avoid confusion with actual probabilities.

The strategy of using heuristic ‘score’ rather than true probabilities has a precedent in the natural language processing literature; for instance, it is employed by Brill (1993, 1995) in his well-known POS tagger. Brill’s tagger makes an initial pass through the document assigning each word the tag most commonly assigned in a training corpus. It then makes a second pass, iteratively applying a set of ‘transformation rules’ of the form ‘if tag $a$ appears adjacent to tag $c$, change tag $a$ to tag $b$’. If multiple rules fire for a given pair of tag $a$ and $w$, the rule which is applied is the one with the highest
‘score’, calculated as follows:

\[
\text{score}(R) = P(b \mid c) - P(a \mid c) = \frac{\text{cnt}(c, b)}{\text{cnt}(c)} - \frac{\text{cnt}(c, a)}{\text{cnt}(c)}
\]  

(4.33)

The ‘score’ of a transformation rule makes reference to probabilities, but the way the tagger algorithm applies rules means it cannot be thought of as ‘computing the most probable set of POS tags’ according to a probability model. It is nonetheless an extremely successful tagger. The probability scores associated with perturbation rules in my model are somewhat similar to the scores associated with transformation rules in Brill’s tagger: though they violate a strict probability model, they are derived from probabilities, and (as I will show later) they work well in practice.

In my proposed model, a calculation of the probability score will be based on the Backoff algorithm as defined in Equations (4.25) - (4.30). If \((w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3)\) is observed in the perturbation corpus, a discount probability score, \(P^*(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3)\) is computed by using Equation (4.26). Otherwise, if \((w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3)\) is not found, the rule is backed off to a lower order model. Backing off to the lower order model (in this case from a trigram to a bigram) means we pick the first two words and the last two words of the rule. Therefore we try to find if there are any occurrences of a bigram rule which is \((w_1 w_{\text{orig}} \rightarrow w_1 w_{\text{pert}})\) or \((w_{\text{orig}} w_3 \rightarrow w_{\text{pert}} w_3)\) in the corpus. If we find it, \(P(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3)\) is estimated by calculating \(P(w_1 w_{\text{orig}} \rightarrow w_1 w_{\text{pert}})\) and \(P(w_{\text{orig}} w_3 \rightarrow w_{\text{pert}} w_3)\). If the bigram rule is unseen, the probability score of \((w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3)\) is estimated using Witten-Bell zero count probability score formula as defined in Equation (4.14).

Hence, the perturbation model is defined below:

\[
P(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3) = \begin{cases} 
P^*(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3), & \text{if } \text{cnt}(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3) > 0 \\
P_{\text{backoff}}(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3), & \text{else if } (\text{cnt}(w_{\text{orig}} w_3 \rightarrow w_{\text{pert}} w_3) + \\
& \text{cnt}(w_1 w_{\text{orig}} \rightarrow w_1 w_{\text{pert}})) > 0 \\
P_{\text{zeropro}}(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3), & \text{otherwise.}
\end{cases}
\]  

(4.34)
If \((w_1 w_{orig} w_3 \rightarrow w_1 w_{pert} w_3)\) is found, its discounted probability score is calculated. This is to set aside some score mass to zero count perturbation rules. A discount probability score, \(P^*()\) is

\[
P^* (w_1 w_{orig} w_3 \rightarrow w_1 w_{pert} w_3) = \frac{\text{cnt}^*_\text{wb}(w_1 w_{orig} w_3 \rightarrow w_1 w_{pert} w_3)}{\text{cnt}(w_1 w_{orig} w_3)} \tag{4.35}
\]

In Equation (4.35), \(\text{cnt}^*\) is computed by using the Witten-Bell discounting equation which has been defined in Equation (4.16) on page 140.

Now, we will see what would happen when \((w_1 w_{orig} w_3 \rightarrow w_1 w_{pert} w_3)\) is unseen in the perturbation corpus. Alternatively, we try to find if there is any occurrences of \((w_1 w_{orig} \rightarrow w_1 w_{pert})\) or \((w_{orig} w_3 \rightarrow w_{pert} w_3)\). If there is, we compute \(P_{\text{backoff}}(w_1 w_{orig} w_3 \rightarrow w_1 w_{pert} w_3)\). Hence, \(P_{\text{backoff}}()\) is defined based on Backoff technique as

\[
P_{\text{backoff}} (w_1 w_{orig} w_3 \rightarrow w_1 w_{pert} w_3) = \frac{1}{2} \times \alpha_{2-\text{left}} P^* (w_{orig} w_3 \rightarrow w_{pert} w_3) + \frac{1}{2} \times \alpha_{2-\text{right}} P^* (w_1 w_{orig} \rightarrow w_1 w_{pert}) \tag{4.36}
\]

In Equation (4.36), weight functions, \(\alpha_{2-\text{left}}\) and \(\alpha_{2-\text{right}}\) are applied to distribute the score mass that was set aside before to the observed bigram rules. Since we take the first two and the last two words in the perturbation rule (the trigram one), the probability score of the left and the right two words must each be halved. Here, both \(\alpha\) functions are defined as follows:

\[
\alpha_{2-\text{left}} = \frac{1 - \sum_{w_{pert} : \text{cnt}(w_1 w_{orig} \rightarrow w_1 [w_{pert}] w_3) > 0} P^* (w_1 w_{orig} w_3 \rightarrow w_1 [w_{pert}] w_3)}{1 - \sum_{w_{pert} : \text{cnt}(w_1 w_{orig} \rightarrow w_1 [w_{pert}] w_3) > 0} P^* (w_1 w_{orig} \rightarrow w_1 [w_{pert}])} \tag{4.37}
\]

and

\[
\alpha_{2-\text{right}} = \frac{1 - \sum_{w_{pert} : \text{cnt}(w_1 w_{orig} w_3 \rightarrow w_1 [w_{pert}] w_3) > 0} P^* (w_1 w_{orig} w_3 \rightarrow w_1 [w_{pert}] w_3)}{1 - \sum_{w_{pert} : \text{cnt}(w_1 w_{orig} w_3 \rightarrow w_1 [w_{pert}] w_3) > 0} P^* (w_{orig} w_3 \rightarrow [w_{pert}] w_3)} \tag{4.38}
\]
Note that \([w_{pert}]\) represents any words available in a corpus. For example, suppose a perturbation rule \((He\ want\ to \rightarrow He\ [w_{pert}]\ to)\) and a list of existing words in the corpus is \((he, want, to, go, book)\). To calculate \(P^*(He\ want\ to \rightarrow He\ [w_{pert}]\ to)\), we replace \(w_{pert}\) with every word in the corpus. Below is the possibility of \(P^*(He\ want\ to \rightarrow He\ [w_{pert}]\ to)\),

\[
\begin{align*}
\text{(32) a. } & P^*(He\ want\ to \rightarrow He\ he\ to) \\
\text{b. } & P^*(He\ want\ to \rightarrow He\ want\ to) \\
\text{c. } & P^*(He\ want\ to \rightarrow He\ to\ to) \\
\text{d. } & P^*(He\ want\ to \rightarrow He\ go\ to) \\
\text{e. } & P^*(He\ want\ to \rightarrow He\ book\ to)
\end{align*}
\]

In the Backoff technique, backing off to lower \(n\)-gram model is continued until one is found (i.e. in this case from a trigram to a bigram model). If a bigram is unseen, we back off to its unigram. However, my proposed perturbation model does not back off to unigrams but instead calculates a zero probability score using the Witten-Bell formula. The justification is as follows. The general objective of my perturbation model is similar to a language model. While a language model is applied to see how likely it is that a certain word occurs in a certain context, the perturbation model is applied to see how likely it is that a certain perturbation is the correction of an erroneous sentence. However, the likely correction model means the perturbation model is strongly expected to propose a grammatical correction, even though the perturbation model doesn’t implement English grammar.

For example, to calculate \(P(He\ want\ to \rightarrow He\ wants\ to)\), if \(\text{cnt}(He\ want\ to \rightarrow He\ wants\ to) = 0\), we can estimate the score if there are any occurrences of \((He\ want \rightarrow He\ wants)\) or \((want\ to \rightarrow wants\ to)\). Else, if \(\text{cnt}(He\ want \rightarrow He\ wants)\) and \(\text{cnt}(want\ to \rightarrow wants\ to)\) is 0, then we need to back off and calculate \(\alpha_{1-gram} P^*(wants \rightarrow want)\). If we back off to unigrams, each word in a corpus will be compared. (Refer to Example (32)). The generation of perturbations featuring every word in the corpus is time consuming. I have tested van der Ham’s model and the processing time is very slow. Furthermore, the performance of my perturbation model is better than his in terms of providing appropriate perturbations and processing time.
Yet, van der Ham’s model generates many ungrammatical perturbations which are unnecessary. Further results of model evaluation are described in §4.6.

With the justification above, if bigram perturbation rules are not found in the corpus, I apply WB zero count probability score algorithm $P_{\text{zeroprob}}$ which is formulated as

$$P_{\text{zeroprob}}(w_1 w_{\text{orig}} w_3 \rightarrow w_1 w_{\text{pert}} w_3) = \frac{T}{Z(T + N)} \quad (4.39)$$

where

- $N$ represents the total number of all perturbation rules (perturbation tokens) in the perturbation corpus.

- $T$ represents how many perturbation rules which have different perturbations (perturbation types) are in the perturbation corpus.

- $Z$ is the number of unseen perturbation rules which is calculated from the difference between the number of possible perturbation rules generated in the corpus, $PR$ and the perturbation types, $T$. All perturbation rules are represented in a trigram model ($AXB \rightarrow AYB$), thus the number of possible rules that can be generated is computed as $PR = V^4$, where $V$ is the vocabulary size of the perturbation corpus. There are $V^2$ possible pairwise combinations of $X$ and $Y$; there are $V$ choices of $A$, and similarly $V$ choices for $B$. These are combined multiplicatively: for each choice of $A$ there are $V$ choices of $B$. Therefore $Z = PR - T$.

### 4.3.2 An Example of The Model

In the following, let me provide a simple example. Suppose an input sentence is *My old is 14* and its perturbed sentence is *My age is 14*. Therefore, I want to calculate what is a probability score of *My age is* given *My old is*, or $P(\text{My old is} \to \text{My age is})$.
The perturbation model is defined as follows:

\[
\begin{align*}
P(my\ old\ is\ →\ my\ age\ is) &= \\
&= \begin{cases} 
P^*(my\ old\ is\ →\ my\ age\ is), & \text{if } cnt(my\ old\ is\ →\ my\ age\ is) > 0 \\
P_{\text{backoff}}(my\ old\ is\ →\ my\ age\ is), & \text{else if } (cnt(my\ old\ →\ my\ age) + \\
&\quad cnt(old\ is\ →\ age\ is)) > 0, \\
P_{\text{zeroprob}}(my\ old\ is\ →\ my\ age\ is), & \text{otherwise.}
\end{cases}
\end{align*}
\]

(4.40)

where

\[
P^*(my\ old\ is\ →\ my\ age\ is) = \frac{cnt^*_{\text{wb}}(my\ old\ is\ →\ my\ age\ is)}{cnt(my\ old\ is)}
\]

(4.41)

and

\[
cnt^*(my\ old\ is\ →\ my\ age\ is) = \\
cnt(my\ old\ is\ →\ my\ age\ is) \times \frac{N(my\ old\ is\ →\ my\ age\ is)}{N(my\ old\ is\ →\ my\ age\ is) + T(my\ old\ is)}
\]

(4.42)

In Equation (4.42) above, the symbol \(N\) represents the total number of perturbation tokens of an observed perturbation rule. Here, where the observed perturbation rule is \((my\ old\ is\ →\ my\ age\ is)\), \(N\) represents how many occurrences of \(my\ old\ is\ →\ my\ age\ is\) are in a perturbation corpus. Meanwhile, \(T\) represents how many perturbation types are available for the observed rule. For instance, how many different perturbed sentences for \((my\ old\ is)\) are present in the perturbation corpus.

\((my\ old\ is\ →\ [\text{different\ perturbations}])\)

To elaborate more on \(T\) and \(N\), suppose a list of perturbation rules is defined in a sample perturbation corpus as shown (33) below:

(33) a. my old is → my age is

b. my old is → my age is
c. my old is → my age is
d. my old is → my age is
e. my old is → my goal is
Based on the corpus (33), there are 4 \((my \ old \ is \rightarrow \ my \ age \ is)\) perturbation tokens in total. This means \(N(my \ old \ is \rightarrow \ my \ age \ is)\) is 4. As for the perturbation types, there are two different interpretations for \(my \ old \ is\) that are \((my \ old \ is \rightarrow \ my \ age \ is)\) and \((my \ old \ is \rightarrow \ my \ goal \ is)\). Thus \(T(my \ old \ is)\) is 2.

If \(cnt(my \ old \ is \rightarrow \ my \ age \ is) = 0\), we try to find if there is any occurrences of \((my \ old \ → \ my \ age)\) or \((old \ is \ → \ age \ is)\). If we do, Equation 4.43 is applied. Since we are computing a backed-off score in two separate ways, we heuristically take the average backed-off score: the sum of each individual score divided by two.

\[
P_{\text{backoff}}(my \ old \ is \rightarrow \ my \ age \ is) = \\
\begin{cases} 
\frac{1}{2} \times \alpha_{2\text{-left}} P^*(old \ is \rightarrow \ age \ is), & \text{if } cnt(old \ is \ → \ age \ is) > 0 \\
+ \\
\frac{1}{2} \times \alpha_{2\text{-right}} P^*(my \ old \ → \ my \ age), & \text{if } cnt(my \ old \ → \ my \ age) > 0 
\end{cases}
\]

(4.43)

The \(\alpha\) functions are used to distribute saved score mass of \((my \ old \ is \ → \ my \ age \ is)\) over \((my \ old \ → \ my \ age)\) and \((old \ is \ → \ age \ is)\). As such, the \(\alpha_{2\text{-left}}\) passes over the score mass for the first two words of the perturbation rule (see Equation 4.44 below). On the other hand, the \(\alpha_{2\text{-right}}\) passes over the score mass for the last two words of the perturbation rule (see Equation 4.45 below).

\[
\alpha_{2\text{-left}} = \frac{1 - \sum_{w_{\text{pert}}: cnt(my \ old \ is \ → \ my \ [w_{\text{pert}}] \ is) > 0} P^*(my \ old \ is \ → \ my \ [w_{\text{pert}}] \ is)}{1 - \sum_{w_{\text{pert}}: cnt(my \ old \ is \ → \ my \ [w_{\text{pert}}] \ is) > 0} P^*(my \ old \ → \ my \ [w_{\text{pert}}])}
\]

(4.44)

\[
\alpha_{2\text{-right}} = \frac{1 - \sum_{w_{\text{pert}}: cnt(my \ old \ is \ → \ my \ [w_{\text{pert}}] \ is) > 0} P^*(my \ old \ is \ → \ my \ [w_{\text{pert}}] \ is)}{1 - \sum_{w_{\text{pert}}: cnt(my \ old \ is \ → \ my \ [w_{\text{pert}}] \ is) > 0} P^*(old \ is \ → \ [w_{\text{pert}}] \ is)}
\]

(4.45)
In Equation 4.44 and 4.45, the \( w_{\text{pert}} \) symbol represents any words which are available in a perturbation corpus.

If no instances of the bigram perturbation rule are found, I directly calculate the Witten-Bell zero count probability score, \( P_{\text{zeroprob}}(\text{my old is} \rightarrow \text{my age is}) \):

\[
P_{\text{zeroprob}}(\text{my old is} \rightarrow \text{my age is}) = \frac{T}{Z(N + T)}
\]  

(4.46)

Based on corpus (33), the total number of all perturbation tokens, \( N \) is 8 and the total number of all perturbation types, \( T \) is 4. Since \( Z = \text{PR} - T \) and the formula of \( \text{PR} \) is \( V^4 \). The \( V \) variable represents the vocabulary size of the corpus (33) (or the total number of different words), so \( V \) is 9. Therefore \( P = 6561 \), \( Z = 6557 \) and \( P_{\text{zeroprob}}(\text{my old is} \rightarrow \text{my age is}) = 0.000051 \).

### 4.4 Development of A N-gram Perturbation Corpus

In this section, I will explain the process of developing a \( n \)-gram perturbation corpus in which the corpus is required to calculate the perturbation scores. Prior to developing the corpus, we need a learner corpus which consists of erroneous sentences and their corresponding corrections of the sentence. Figure 4.3 shows a process flow of developing the \( n \)-gram perturbation corpus.

Referring to Figure 4.3, there are two processes: the creation of a sentence perturbation corpus from a learner corpus and the creation of a \( n \)-gram perturbation corpus from the sentence perturbation corpus. The learner corpus is a collection of learners’ grammatical and ungrammatical sentences. The learner corpus that I used is the same corpus that I gathered before as previously described in Chapter 3. The sentence perturbation corpus consists of a list of ungrammatical sentences and their proposed corrections which I extract from the learner corpus. I name each item in this list, a sentence perturbation. Suppose an example of a small subset of the sentence perturbation corpus is shown in (34) below.

\[(34) \quad \text{a.} \quad (\text{“I from Malaysia”, “I am from Malaysia”})\]
Each sentence perturbation in (34) comprises a pair of sentences, of which the first represents a learner’s original (erroneous) sentence and the second is a corrected version of this sentence. If the original sentence has more than one possible correction, it is represented in more than one sentence perturbation e.g. in (34e) and (34f).

Next, from the sentence perturbation corpus, I create a $n$-gram perturbation corpus. An algorithm to create the corpus is presented in Algorithm 1. The algorithm is a version of the Levenshtein edit-distance algorithm (Levenshtein, 1966). Three models of $n$-gram perturbation corpus are generated: trigram, bigram and unigram perturbation corpus. First, the trigram perturbation corpus is created and then the bigram and unigram perturbation corpus.

![Figure 4.3: Process for Development of a $N$-gram Perturbation Corpus](image)
Algorithm 1 Create N-gram Perturbation Corpus

1: for each learner data in learner corpus do
2:    sentc_pert ← (original_sente, corrected_sente)
3:    append sentc_pert to sentc_pert_corpus
4: end for
5: for each sentc_pert in sentc_pert_corpus do
6:    trigram_pert_corpus ← Make-trigram-perturbations for sentc_pert
7:    bigram_unigram_pert_corpus ← Make-bigram-and-unigram-perturbations from trigram_pert_corpus
8: end for
9: merge trigram_pert_corpus and bigram_unigram_pert_corpus into ngram_perturbation_corpus

Similar to a sentence perturbation corpus, the n-gram perturbation corpus consists of a list of pairs of n-gram perturbations. Each pair consists of an original n-gram and a corrected n-gram. In a trigram model, the general structure of the original trigram is represented as a set of three words, ("w₁ w₂ₜᵣᵢ₉₉₉₃") and its corrected trigram as ("w₁ wₚₑᵣ₉₉₉₃"). As such, the respective trigram perturbation is represented as ("w₁ wₘᵢ₉₉₉₃", "w₁ wₚₑᵣ₉₉₉₃"). The w₁ and w₃ in the original and corrected n-gram are the same word respectively. The w₉ᵣᵢ₉₉₉₃ and wₚₑᵣ₉₉₉₃ are words which can be represented by a word or a blank space. The blank space symbol is (gap). Yet, both w₉ᵣᵢ₉₉₉₃ and wₚₑᵣ₉₉₉₃ are surrounded by asterisk symbols (*) to show that w₉ᵣᵢ₉₉₉₃ has been perturbed to wₚₑᵣ₉₉₉₃. The wₚₑᵣ₉₉₉₃ is also known as a perturbed word. This means one of a perturbation action (insert/delete/substitute/transposition) is applied to the perturbed word. For instance, from a sentence perturbation 

("I from Malaysia", "I am from Malaysia")

we can generate a trigram perturbation such as

(35) ("I *_gap_* from", "I *am* from")

Here what we can derive from (35) is, in order to perturb “I from Malaysia” to “I am from Malaysia”, a word insertion is applied in which we insert a word “am” between
“I” and “from”. Further explanation about the creation of trigram perturbations is described in §4.4.1.

van der Ham (2005) has already developed a function to generate a $n$-gram perturbation corpus in the Kaitito system. However I find two limitations of the function. Firstly, the function only considers the first error that it finds. If the sentence has more than one error, only the first located error is counted and the remaining errors are ignored. For instance, from a sentence perturbation

(36) ("He sales many fishes at the market", “He sells many fish at the market")

two trigram perturbations that can be generated are

(37) a. (“He *sales* many”, ”He *sells* many”)
    b. (“many *fishes* at”, ”many *fish* at”)

But in van der Ham’s function, only (37a) is generated. This means only one trigram perturbation can be generated from one sentence perturbation. As such, we can’t generate many trigram perturbations which can be useful in suggesting corrections for erroneous sentences. Therefore, I develop a function which is able to generate several trigram perturbations from a sentence perturbation which has more than one error. For example, my function is able to generate (37a) and (37b) from (36).

The second limitation is the sentence perturbation corpus such as in (34) has to be manually typed and saved in a computer file format which the function is able to access in the file. Due to the large amount of data in my learner corpus, a manual typing job into a file that the function understands is a very tedious and time-consuming job. Since my learner corpus is stored in the Excel file format, I created a new function that is able to automatically read all data from the Excel file and generate a sentence perturbation corpus. After that I created another function to generate a $n$-gram perturbation corpus from the sentence perturbation corpus.

### 4.4.1 Generating trigram perturbations

A $n$-gram perturbation corpus consists of a list of $n$-gram perturbations. The types of $n$-gram perturbation corpus created are trigram, bigram and unigram models. As
I mentioned earlier, the trigram perturbation corpus is created first, followed by the bigram and unigram perturbation corpus. Algorithm 2 describes step by step instructions on how trigram perturbations are created. The creation process starts by locating the first location of an unmatched word between an original sentence and its corrected sentence of a sentence perturbation. Then, until the original sentence is equivalent to its corrected sentence, one of four functions will be performed. The functions are **word-insertion**, **word-deletion**, **word-substitution**, and **word-transposition**. Each process will be performed if a specified condition/s for each process is/are met. In the following, I will describe how each process is performed together with an example for each process.

Suppose a sentence perturbation corpus has four sentence perturbations as shown in (38).

(38)  
   a. (“I from Malaysia”, “I am from Malaysia”)  
   b. (“I Malaysia”, “Malaysia”)  
   c. (“He sales many fishes at the market”, “He sells many fish at the market”)  
   d. (“I to go school”, “I go to school”)  

First of all, each original and corrected sentence in (38) will be preceded by _sos_ and ended by _eos_ symbols. Both symbols are indicator symbols which _sos_ denotes *the start of a sentence* and _eos_ denotes *the end of a sentence*. An example in (39) is the result after inserting the indicator symbols in (38).

(39)  
   a. (“_sos_ I from Malaysia _eos_”, “_sos_ I am from Malaysia _eos_”)  
   b. (“_sos_ I Malaysia _eos_”, “_sos_ Malaysia _eos_”)  
   c. (“_sos_ He sales many fishes at the market _eos_”, “_sos_ He sells many fish at the market _eos_”)  
   d. (“_sos_ I to go school _eos_”, “_sos_ I go to school _eos_”)  

Each sentence perturbation in (39) represents how one of 4 perturbation actions is executed. An example (39a) will show how word-insertion is done, (39b) involves word-deletion, (39c) and (39d) respectively involve word-substitution and word-transposition.
Algorithm 2 Make-trigram-perturbations for \((\text{original}, \text{corrected})\)

Require: \(\text{original} = (w_1 w_2 w_3 \ldots w_n)\) and \(\text{corrected} = (w_1 w_2 w_3 \ldots w_m)\)

1: locate the first unmatched word, \(\text{loc}_\text{unmtched}\) between \(\text{original}\) and \(\text{corrected}\)

2: while \(\text{original} \neq \text{corrected}\) do

3: \(\text{error}\# \leftarrow \text{the number of unmatched words between } \text{original} \text{ and } \text{corrected}\)

4: \(\text{cur}_\text{loc} \leftarrow \text{loc}_\text{unmtched}\)

5: if \(((w_{(\text{cur}_\text{loc})} \text{ in } \text{original} = w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{corrected}) \text{ AND } (w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{original} \neq w_{(\text{cur}_\text{loc})} \text{ in } \text{corrected})) \text{ then}\)

6: \(\text{trigram}\_\text{pert} \leftarrow \text{Word-insertion at location } \text{cur}_\text{loc}\)

7: else if \((w_{(\text{cur}_\text{loc})} \text{ in } \text{original} = w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{corrected}) \text{ then}\)

8: \(\text{trigram}\_\text{pert} \leftarrow \text{Word-deletion at location } \text{cur}_\text{loc}\)

9: else if \((w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{original} = w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{corrected}) \text{ then}\)

10: \(\text{trigram}\_\text{pert} \leftarrow \text{Word-substitution at location } \text{cur}_\text{loc}\)

11: else if \(((w_{(\text{cur}_\text{loc})} \text{ in } \text{original} = w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{corrected}) \text{ AND } (w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{original} = w_{(\text{cur}_\text{loc})} \text{ in } \text{corrected})) \text{ then}\)

12: \(\text{trigram}\_\text{pert} \leftarrow \text{Word-transposition at location } \text{cur}_\text{loc}\)

13: else if \(((w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{original} \neq w_{(\text{cur}_\text{loc}+1)} \text{ in } \text{corrected}) \text{ AND } (\text{error}\# \leq 2)) \text{ then}\)

14: \(\text{original} \leftarrow \text{the current } \text{original} \text{ plus a replacement of } w_{(\text{cur}_\text{loc})} \text{ in } \text{original} \text{ to } w_{(\text{cur}_\text{loc})} \text{ in } \text{corrected}\)

15: end if

16: locate \(\text{loc}_\text{unmtched}\) between \(\text{original}\) and \(\text{corrected}\)

17: end while

18: append \(\text{trigram}\_\text{pert}\) to \(\text{trigram}\_\text{pert}\_\text{list}\)

19: return \(\text{trigram}\_\text{pert}\_\text{list}\)
4.4.1.1 Word-insertion function

In (39a), the original sentence is

“\_sos\_ I from Malaysia \_eos\_”

and the corrected sentence is

“\_sos\_ I am from Malaysia\_eos\_”

Algorithm 2 begins with locating the first unmatched word between the original and corrected sentence. Suppose, the symbol “\_sos\_” is the location of the first word, thus the location of unmatched word is 3. In Algorithm 2, a variable cur\_loc represents the location of unmatched word, so cur\_loc = 3.

Original sentence: "\_sos\_ I from Malaysia \_eos\_"

unmatched word matched word

Corrected sentence: "\_sos\_ I am from Malaysia \_eos\_"

Figure 4.4: Location of unmatched and matched words at cur\_loc in the original sentence

It indicates the third word in the original sentence does not match with the third word in the corrected sentence. However, the third word (at cur\_loc) in the original sentence matches to the fourth word (at cur\_loc+1) in the corrected sentence. Figure 4.4 graphically shows the matched and unmatched word at cur\_loc between the original and corrected sentence.

The previously mentioned situation satisfies a condition stated in line 5 of Algorithm 2. As such, the word-insertion function is executed. Algorithm 3 demonstrates how the function works. Initially the function starts by inserting a symbol “\_gap\_” at cur\_loc in the original sentence. Then the symbol and the unmatched word at cur\_loc
in the corrected sentence is marked with an asterisk symbol, “∗”. An output of the function is a new trigram perturbation which consists of a pair of an original trigram and a corrected trigram. The pair consists of 3 words and only the middle word of the original and corrected trigram are unequal. The middle word of the original trigram is “_gap_” and the middle word of the corrected trigram is the unmatched word in the corrected sentence. Therefore, the original trigram consists of the word in the original sentence at \textit{cur.loc-1}, followed by a “_gap_” symbol, and then a word at \textit{cur.loc} in the original sentence as shown in line 4 of Algorithm 3. The corrected trigram comprises the words in the corrected perturbation at \textit{cur.loc-1}, \textit{cur.loc}, and \textit{cur.loc+1} locations. Consequently, the new trigram perturbation generated is as follows:

(40) (“I ∗_gap_∗ from”, “I ∗am∗ from”)

The word-insertion function ends by updating the original sentence by inserting the unmatched word in the corrected sentence (“am”) at location \textit{cur.loc}. Then the function goes back to the make-trigram-perturbation function with the updated original sentence.

\begin{algorithm}
\caption{Word-insertion at location \textit{cur.loc}}
\begin{algorithmic}[1]
\State insert a “_gap_" string at location \textit{cur.loc} in original
\State \textit{ins.wrd} ← a word at \textit{w(cur.loc)} in corrected
\State mark “_gap_" and \textit{ins.wrd} with “∗"
\State \textit{trigram.orig} ← \textit{(w(cur.loc-1) ∗_gap_∗ w(cur.loc))} in original
\State \textit{trigram.crtd} ← \textit{(w(cur.loc-1) ∗ins.wrd∗ w(cur.loc+1))} in corrected
\State \textit{trigram.pert} ← (\textit{trigram.orig}, \textit{trigram.crtd})
\State update original by inserting \textit{ins.wrd} at \textit{cur.loc}
\State \textbf{return} \textit{trigram.pert}
\end{algorithmic}
\end{algorithm}

4.4.1.2 Word-deletion function

The sentence perturbation (39b) is (“\_sos\_ I Malaysia _eos_”, “\_sos\_ Malaysia _eos_”). When Algorithm 2 is gone through, \textit{cur.loc} is 2. A condition in line 7 of Algorithm 2 is fulfilled. This means the word at \textit{cur.loc} in the original sentence matches with the
word at \texttt{cur\_loc+1} in the corrected sentence. Therefore, the word-deletion function is performed. Algorithm 4 describes how the word-deletion function works. In contrast with the word-insertion function, a “\texttt{gap}.” symbol is inserted in the corrected sentence. The new trigram perturbation generated is

\begin{equation}
(41) \quad (“\texttt{sos\_} *I* Malaysia”, “\texttt{sos\_} *\texttt{gap\_} * Malaysia”)
\end{equation}

The word-deletion function ends by updating the corrected sentence by deleting the unmatched word in the original sentence (“I”) at location \texttt{cur\_loc}. Then the function goes back to the make-trigram-perturbation function with the updated original sentence.

\begin{algorithm}[H]
\caption{Word-deletion at location \texttt{cur\_loc}}
\begin{algorithmic}[1]
\State Insert “\texttt{gap}.” at location \texttt{cur\_loc} in \texttt{corrected}
\State \texttt{del\_wrd} $\leftarrow$ \texttt{w}(_{\texttt{cur\_loc}}) in \texttt{original}
\State Mark “\texttt{gap}.” and \texttt{del\_wrd} with “*”
\State \texttt{trigram\_crtd} $\leftarrow$ (\texttt{w}(_{\texttt{cur\_loc}-1}) “* \texttt{gap\_} *” \texttt{w}(_{\texttt{cur\_loc}})) in \texttt{corrected}
\State \texttt{trigram\_orig} $\leftarrow$ (\texttt{w}(_{\texttt{cur\_loc}-1}) \texttt{* del\_wrd} \texttt{* w}(_{\texttt{cur\_loc}+1})) in \texttt{original}
\State \texttt{trigram\_pert} $\leftarrow$ (\texttt{trigram\_orig}, \texttt{trigram\_crtd})
\State Update \texttt{original} by deleting \texttt{w}(_{\texttt{cur\_loc}})
\State \textbf{return} \texttt{trigram\_pert}
\end{algorithmic}
\end{algorithm}

4.4.1.3 Word-substitution function

The word-substitution function is performed while generating trigram perturbations from the sentence perturbation (39c). The original sentence is

“\texttt{sos\_} He sales many fishes at the market \texttt{eos\_}”

and the corrected sentence is

“\texttt{sos\_} He sells many fishes at the market \texttt{eos\_}”

In (39c), the location of first different word, \texttt{cur\_loc} is 3. In this situation, a word at \texttt{cur\_loc+1} in both original and corrected sentence is equal, so this satisfies a condition in line 9 of Algorithm 2. An algorithm for the word-substitution function is highlighted
in Algorithm 5. After executing the algorithm, a newly created trigram perturbation is

\[(42) \left( \text{“He *sales* many”}, \text{“He *sells* many”} \right) \]

Then, the original sentence is updated by replacing the word \textit{sales} with the word \textit{sells} in the corrected sentence, which yields an updated sentence perturbation as in 43 below:

\[(43) \text{“sos. He sells many fishes at the market _eos_.”}, \text{“sos. He sells many fish at the market _eos_”} \]

The word-substitution function ends by returning back to the make-trigram-perturbation function with the updated original sentence. However, the original sentence and the corrected sentence in (43) are still not equal since the first unmatched word location is found in (43). Here, \textit{cur-loc} is 5. The word-substitution is again executed as the same condition in line 9 of Algorithm 2 is again satisfied. Then another trigram perturbation is generated which is,

\[(44) \left( \text{“many *fishes* at”}, \text{“many *fish* at”} \right) \]

After replacing the word \textit{fishes} to \textit{fish} in the original sentence, finally the original and corrected sentence are matched. As such, no more trigram perturbations will be generated.

\begin{algorithm}
\caption{Word-substitution at location \textit{cur-loc}}
\begin{algorithmic}[1]
\State $\textit{orig\_subst\_wrd} \leftarrow w_{\text{(cur\_loc)}}$ in \textit{original}
\State $\textit{crtd\_subst\_wrd} \leftarrow w_{\text{(cur\_loc)}}$ in \textit{corrected}
\State mark $\textit{orig\_subst\_wrd}$ and $\textit{crtd\_subst\_wrd}$ with "*"
\State $\textit{trigram\_orig} \leftarrow \left( w_{\text{(cur\_loc-1)}} \ast \textit{orig\_subst\_wrd} \ast w_{\text{(cur\_loc+1)}} \right)$ in \textit{original}
\State $\textit{trigram\_crtd} \leftarrow \left( w_{\text{(cur\_loc-1)}} \ast \textit{crtd\_subst\_wrd} \ast w_{\text{cur\_loc+1}} \right)$ in \textit{corrected}
\State $\textit{trigram\_pert} \leftarrow \left( \textit{trigram\_orig}, \textit{trigram\_crtd} \right)$
\State update \textit{original} by replacing $\textit{orig\_subst\_wrd}$ at \textit{cur\_loc} with $\textit{crtd\_subst\_wrd}$
\State \textbf{return} $\textit{trigram\_pert}$
\end{algorithmic}
\end{algorithm}
4.4.1.4 Word-transposition function

A sentence perturbation (39d), (“sos_ I to go school _eos_”, “sos_ I go to school _eos_”), involves a word-transposition function\(^4\) to be executed. Algorithm 6 explains how the function works. In (39d), the first unmatched word location between the original and corrected sentence, \(cur_{loc} = 3\). As referred to (39d), the word at \(cur_{loc}\) in the original sentence is equal to the word at \(cur_{loc}+1\) in the corrected sentence, and vice-versa. As such a condition in line 11 of Algorithm 2 is met which causes the word-transposition to be performed. The word-transposition is similar to the word-substitution but in the word-transposition, two words are involved instead. An alternative way to the word-transposition function is performing the word-substitution twice. However, I prefer to use the word-transposition function as it only involves swapping two adjacent words to fix two errors. A new trigram perturbation generated is

\[(45) \ ("I *to go* school", "I *go to* school")\]

Here the trigram perturbations generated from the word-transposition function consist of 4 words instead of 3. The word-transposition function ends by updating the original sentence by substituting its two words at \(cur_{loc}\) and \(cur_{loc}+1\) locations with two words at the same location in the corrected sentence ("go to"). Lastly, the function ends by returning back to the make-trigram-perturbation function with the updated original sentence.

4.4.2 Adjacent Errors

The simplest errors to extract are cases when a sentence only contains a single error. In cases where there are two errors in a single sentence, it is somewhat harder to identify appropriate trigrams. I implement two alteration policies in such cases. One policy is to ignore errors when they are close together. The other is to try to extract appropriate trigrams. The latter policy will yield more training data for the perturbation model, but it will probably be of slightly lower quality. The latter policy is implemented by a condition on line 13 of Algorithm 2. Suppose a sentence perturbation:

\(^4\)Word-transposition perturbation is not handled by my proposed error correction model. This is a topic for future work.
Algorithm 6 Word-transposition at location $cur\_loc$

1: $orig\_transp\_wrd \leftarrow (w_{(cur\_loc)} \cdot w_{(cur\_loc+1)})$ in original

2: $crtd\_transp\_wrd \leftarrow (w_{(cur\_loc)} \cdot w_{(cur\_loc+1)})$ in corrected

3: mark $orig\_transp\_wrd$ and $crtd\_transp\_wrd$ with "*

4: $trigram\_orig \leftarrow (w_{(cur\_loc-1)} \cdot orig\_transp\_wrd \cdot w_{(cur\_loc+2)})$ in original

5: $trigram\_crtd \leftarrow (w_{(cur\_loc-1)} \cdot crtd\_transp\_wrd \cdot w_{cur\_loc+2})$ in corrected

6: $trigram\_pert \leftarrow (trigram\_orig, trigram\_crtd)$

7: update $original$ by replacing $orig\_transp\_wrd$ at $cur\_loc$ with $crtd\_transp\_wrd$

8: $return$ $trigram\_pert$

(46) ("He like to reading", "He likes reading")

In order to generate trigram perturbations from (46), I have to amend (46) in such a way that there is only one unmatched word between its original and corrected sentence. By substituting the first unmatched word in the original sentence with the first unmatched word in the corrected sentence, the new sentence perturbation now becomes:\footnote{Of course, this policy does not always work: it is a heuristic, which derives more data from the corpus at the expense of data quality. A case where it would not work is $He$ to like reading, which would be transformed to $He$ likes like reading.}

(47) ("He likes to reading", "He likes reading")

When (47) is passed to Algorithm 2, a new trigram generated is

(48) ("likes *to* reading", "likes *_gap_* reading")

4.4.3 Generating bigram and unigram perturbations

In §4.4.1, a trigram perturbation corpus has been generated and is shown in (49) below.

(49) a. ("I *_gap_* from", "I *am* from")

b. ("sos_ *I* Malaysia", "sos_ *_gap_* Malaysia")

c. ("He *sales* many", "He *sells* many")

d. ("many *fishes* at", "many *fish* at")

e. ("I *to go* school", "I *go to* school")
Based on Algorithm 1 for creating a \( n \)-gram perturbation corpus, after a trigram perturbation corpus is created, a bigram and a unigram perturbation corpus are generated (refer to line 7 of Algorithm 1). Generation of the bigram and unigram perturbation corpus is described in Algorithm 7. For each trigram perturbation in the trigram perturbation corpus, two bigram perturbations and one unigram perturbation are generated (from line 3 until line 10). The bigram corpus consists of a list of bigram perturbations generated from trigram perturbations. Both bigram perturbations must include the perturbed word, and the unigram perturbation consists only of the perturbed word. For example, from (49a), two bigram perturbations that are generated are

\[(50)\quad \begin{align*}
a. & \quad ("I \ast\text{gap}\ast", "I \ast\text{am}\ast") \\ b. & \quad ("\ast\text{gap}\ast \text{ from}", "\ast\text{am}\ast \text{ from}")
\end{align*}\]

and a unigram perturbation is

\[(51)\quad ("\ast\text{gap}\ast", "\ast\text{am}\ast")\]
4.4.4 Counting $n$-gram perturbations

In the $n$-gram perturbation corpus, I also keep a record of how many times each $n$-gram perturbation occurs. The example of the $n$-gram perturbation corpus together with its occurrence number is shown in (52).

$$(52) \quad \begin{align*}
  a. & \quad (("I * \_gap_* from", "I *am* from"), 1) \\
  b. & \quad (("I * \_gap_*", "I *am*"), 1) \\
  c. & \quad ((" * \_gap_* from", "*am* from"), 1) \\
  d. & \quad (("* \_gap_*", "*am*"), 1) \\
  e. & \quad (("sos_*I* Malaysia", "sos_* \_gap_* Malaysia"), 1) \\
  f. & \quad (("sos_*I*", "sos_* \_gap_*"), 1) \\
  g. & \quad (("*I* Malaysia", "* \_gap_* Malaysia"), 1) \\
  h. & \quad (("I", "* \_gap_*"), 1) \\
  i. & \quad (("He *sales* many", "He *sells* many"), 1) \\
  j. & \quad (("He *sales*", "He *sells*"), 1) \\
  k. & \quad (("*sales* many", "*sells* many"), 1) \\
  l. & \quad (("sales", "sells"), 1) \\
  m. & \quad (("many *fishes* at", "many *fish* at"), 1) \\
  n. & \quad (("many *fishes*", "many *fish*"), 1) \\
  o. & \quad (("*fishes* at", "*fish* at"), 1) \\
  p. & \quad (("*fishes", "*fish"), 1) \\
  q. & \quad (("I *to go* school", "I *go to* school"), 1) \\
  r. & \quad (("I *to go*", "I *go to*"), 1) \\
  s. & \quad (("*to go* school", "*go to* school"), 1) \\
  t. & \quad (("*to go*", "*go to*"), 1)
\end{align*}$$

If there is another trigram perturbation say ("I * \_gap_* from", "I *am* from") added into (40), the frequency number of (52a), (52b), (52c), and (52d) is incremented, as depicted below.
4.5 Generating feasible perturbations

The $n$-gram perturbation corpus created in §4.4 can now be used as a reference to perform word-level perturbations of an input sentence. In the Kaitito dialogue system, every time an input sentence that cannot be parsed by Kaitito parsing system is presented, the system assumes the sentence is ungrammatical. Then the sentence will undergo a word-level perturbation in order to produce a list of candidate sentences. The four perturbation actions involved in producing these candidates are word-deletion, word-insertion, word-substitution, and word-transposition.

Some may notice that the names of these four actions are mentioned before in §4.4.1 and wonder if there is any differences between the four mentioned here and in §4.4.1. Generally, these four actions work similarly to the four functions described in §4.4.1, but the only difference is these four perturbation actions are applied when generating feasible perturbations for a given input sentence. While generating the feasible perturbations, the $n$-gram perturbations which have been generated by the four functions described in §4.4.1 are referred to. Therefore, the four actions mentioned here and the four functions mentioned in §4.4.1 are executed at different times. Next I will explain each of the former actions and how it is applied to an input sentence. I use the $n$-gram perturbation corpus in (52) as a reference during the explanation.

4.5.1 Insertion

For each word in an input sentence, say $wrd$, if there is an occurrence of the word in a $n$-gram perturbation corpus and $wrd$ is preceded or succeeded by a gap (_gap_) in any original $n$-gram, then a word-insertion is applied. Suppose the input sentence is “He from Malaysia” and then it becomes “_sos_ He from Malaysia _eos_”. Based on (52c)
which is

\( (\text{"_gap_}* \text{from"}, \text{"am* from"}) \)

a feasible perturbation for “He from Malaysia” is

\( (\text{"He *gap* from Malaysia"}, \text{"He *am* from Malaysia"}) \)

### 4.5.2 Deletion

A word-deletion perturbation works in contrast with the word-insertion perturbation. Instead of finding any occurrences of a word that is preceded or succeeded by a \text{.gap} in any original \text{n}-gram in a \text{n}-gram perturbation corpus, a \text{.gap} in any corrected \text{n}-gram is looked for. For instance, an input sentence is "I Indonesia". The perturbation data (52f) which is

\( (\text{"_sos_*l*"}, \text{"_sos_*_gap_*"}) \)

is used to make a word-deletion perturbation that is

\( (\text{"*l* Indonesia"}, \text{"*_gap_* Indonesia"}) \)

### 4.5.3 Substitution

For each word in an input sentence, say \text{wrd} and for each \text{n}-gram perturbation in the \text{n}-gram perturbation corpus, say \text{pert}, if there are any occurrences of \text{wrd} in an original \text{n}-gram of \text{pert} which is substituted to another word (except a \text{.gap}) in a corrected \text{n}-gram of \text{pert}, then a word-substitution is applied. In this case, a unigram perturbation corpus is normally consulted. For instance, a possible word-substitution perturbation for “(I have fishes today)” is

\( (\text{"I have *fishes* today"}, \text{"I have *fish* today"}) \)

based on the unigram perturbation (52p), (\text{"*fishes"}, \text{"*fish*"}).
4.5.4 Transposition

A *word-transposition* works similarly to a *word-substitution* in such a way that the *word-transposition* involves two words instead of one word. For each pair of words in an input sentence, say $w_1w_2$ and for each $n$-gram perturbation in the $n$-gram perturbation corpus, say pert, if there are any occurrences of $w_1w_2$ in an original $n$-gram of pert, it can be replaced with $w_2w_1$ (except a *gap* in either $w_1$ and $w_2$) in a corrected $n$-gram of pert, then a *word-transposition* is applied. Similar to the word-substitution case, a unigram perturbation corpus is also normally looked up in the word-transposition. For instance, for a sentence perturbation “(They to go school)”, a possible *word-transposition* perturbation is

(57) (“They *to go* school”, “They *go to* school”)

based on the unigram perturbation (52t) which is (“*to go*”, “*go to*”).

4.6 An Evaluation of the Proposed Model

In this section, I evaluate the statistical model of error correction which I have described in §4.3. An objective of the evaluation is to examine how effectively the model can propose appropriate and grammatical corrections for an ungrammatical input sentence. Firstly, I start with a methodology of the evaluation which includes the error correction model, data sets used, parameters defined during the evaluation, and how the evaluation is carried out. Then I will present the results produced from the evaluation.

4.6.1 The Model to be Evaluated

An evaluated model is a statistical model of error correction which was already described in detailed in §4.3. The model is developed based on Backoff and Witten-Bell discounting techniques. The definition of the statistical model is contained in Formula 4.34 until Formula 4.39 on page 148.
4.6.2 Data

Training data and test data sets are created from a sentence perturbation corpus. The sentence perturbation corpus is extracted from a learner corpus that I gathered before, as previously explained in Chapter 3. The creation of the sentence perturbation corpus was explained in §4.4.

4.6.3 Parameters

I implemented different versions of the sentence perturbation corpus using different parameter settings. The parameters are distinguished by various error models and types of training data sets. Each error model is distinguished by a different scenario as described next, and the training data types will be explained in §4.6.3.4.

4.6.3.1 Error model 1: Comprehensive

This error model uses the full learner corpus from which all spelling errors have been removed. My model is set up to correct grammatical or language errors, not spelling errors or typos.

4.6.3.2 Error model 2: PN-label-backoff

This error model is the same as the Comprehensive model but with all proper names changed to the \(<\text{pn}>\) symbol. For instance, a sentence perturbation

\[ ("I from Malaysia", "I am from Malaysia") \]

is edited to

\[ ("I from <\text{pn}>", "I am from <\text{pn}>") \]

This model is used to address a data sparseness problem in Comprehensive. Proper names can lead to the problem of sparse data. They are huge in numbers and it is very hard to include all possible proper names in a vocabulary. For example, for two different sentence perturbations as follows

\[ \text{(58) a. ("I was born Kuala Lumpur", "I was born in Kuala Lumpur") } \]
b. (“I was born Melaka”, “I was born in Melaka”)

In PN-label-backoff, the sentence perturbations in (58) become two similar sentence perturbations as shown in (59).

(59) a. (“I was born <pn>”, “I was born in <pn>”)
   b. (“I was born <pn>”, “I was born in <pn>”)

PN-label-backoff is a model with fewer sparse data problems. An evaluation result for this model will be compared to the evaluation result for Comprehensive. The results will be discussed in §4.7.

4.6.3.3 Building separate error models for different sentence types

I then create several error models for specific types of sentence. The errors found in a sentence may be relative to the type of sentence it is, for instance whether it is present or past tense. While these specific models have less data, the data should be of a higher quality. This may result in an improved model overall.

Several error models are built to represent different types of sentence. While PN-label-backoff is a model which has less sparse data problem, each error model built here consists of different sentence types which represent a specific type context. The contexts are distinguished by tense forms, types of pronoun, and types of posed question.

Error model 3: Tense I built three error models, by dividing the Comprehensive model into three subsets: Present-tense, Past-tense and Future-tense.

Error model 4: Pronoun I built four error models, by dividing Comprehensive into four subsets. These four error models are categorised by a person type and a grammatical number. The type of person is either 1st person or 3rd person and the grammatical number is either Singular or Plural. Table 4.3 lists the four error models as well as the pronoun/s used in sentences for each sub-model.
Table 4.3: Pronoun types

<table>
<thead>
<tr>
<th></th>
<th>Plural</th>
<th>Singular</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st person</td>
<td>we</td>
<td>I</td>
</tr>
<tr>
<td>3rd person</td>
<td>they, my parents</td>
<td>he, she, my best friend</td>
</tr>
</tbody>
</table>

Table 4.4: Question types

<table>
<thead>
<tr>
<th>Wh-questions</th>
<th>Open-ended questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Which city are you from?</td>
<td>Tell me about your city.</td>
</tr>
<tr>
<td>What did your best friend do last weekend?</td>
<td>Describe your parents.</td>
</tr>
<tr>
<td>What will your parents do this evening</td>
<td>Describe what your father does in his job.</td>
</tr>
</tbody>
</table>

Error model 5: Question Types I built four error models, by dividing *Comprehensive* into two subsets. The first subset consists of sentences or responses from Wh-question types (*Wh-q*) and the second one consists of open-ended question responses (*Open-q*). Table 4.4 shows some samples of questions for both types.

4.6.3.4 Mining errors from the learner corpus: Training data sets

Types of training data are differentiated by how many n-gram perturbations are generated. There are three types of training data set as listed below.

**Type 1** This training data set consists of n-gram perturbations generated from original sentences of sentence perturbations which contain one error only. These are likely to be the most accurately identified error n-gram perturbations, as argued in §4.4.1.

**Type 2** This training data set consists of n-gram perturbations generated from original sentences of sentence perturbations which have one or more errors, but *excluding* errors that are adjacent to each other. These are likely to be a little less accurate but the training set will be larger.

**Type 3** This training data set consists of n-gram perturbations generated from original sentences of sentence perturbations which have have one or more errors, *including* errors that are adjacent to each other. These are likely to be the least accurately
identified error $n$-gram perturbations. Please refer back §4.4.2 in order to see how I generate trigram perturbations from multiple error sentence perturbations which the errors are adjacent to each other.

Let me give an example of the contents in Type 1, Type 2 and Type 3 respectively. Suppose a sentence perturbation corpus consists of the following sentence perturbations

\begin{enumerate}
\item[60] a. ("I from Malaysia", "I am from Malaysia")
\item b. ("I Malaysia", "Malaysia")
\item c. ("He sales many fishes at the market", "He sells many fish at the market")
\item d. ("I to go school", "I go to school")
\item e. ("He like to reading", "He likes reading")
\item f. ("He like to reading", "He likes to read")
\end{enumerate}

For each sentence perturbation in (60), numbers of errors are determined by calculating the difference between an original and a corrected sentence. Here, I adapt the Levenshtein distance algorithm (Levenshtein, 1966). The algorithm measures the amount of difference between two strings or sentences. Based on (60), only (60a) and (60b) consist of one error. The remaining item have more than one error. Therefore, only (60a) and (60b) are chosen for generating $n$-grams perturbations for Type 1 training data set.

The sentence perturbations (60c), (60e), and (60f) consist of more than one error but the last two have errors which are adjacent to each other. As such, for Type 2 training data, (60a) to (60d) are considered. A reason why (60d) is taken for Type 2 is because a correction can be done by performing a transformation of the two adjacent words. As for Type 3, all errors in (60) are counted.

Obviously, Type 1 is a subset of Type 2 and Type 2 is a subset of Type 3. This means the number of $n$-gram perturbations in Type 1 is less than in Type 2, and the number of $n$-gram perturbations in Type 2 is less than in Type 3.
4.6.4 Evaluation Procedure

I evaluate the statistical model of error correction using a \( n \)-fold cross-validation technique. This technique can be used to evaluate how effectively the statistical model can propose appropriate and grammatical corrections for an ungrammatical input sentence based on some empirical collected data. In this technique, each segment of data becomes a training set and a data set (Manning and Schütze, 1999). The \( n \)-fold indicates how many \( n \) segments the data is partitioned into. Then, the model will be evaluated on \( n \) rounds. For each round, there are three processes involved: \textit{data partitioning}, \textit{data analysis} and \textit{data validation}. Data is partitioned into two subsets: a data test and a training data set. Data analysis is performed to analyse the training data set. Lastly, an error model is validated based on the analysis outcome.

4.6.4.1 Data Partitioning

The sentence corpus described in §4.6.2 is used as the data. The data is categorised by various error models as described in §4.6.3. During the \textit{data partitioning} process, each error model is partitioned into five segments, so \( n \) is 5. For each evaluation round, four segments are assigned as the training set and a remaining one segment as a test set. For each round, a different segment is assigned as test data. For example, in the first round, the first four segments are assigned as the training set and the last segment as the test set. In the second round, the second segment is assigned as the test data and the rest becomes the training set, and so forth.

4.6.4.2 Data Analysis

In the \textit{data analysis} process, a \( n \)-gram perturbation corpus is generated from each training data set. The generation of the \( n \)-gram perturbation corpus has already been described in-detail in §4.4 on page 154. The \( n \)-gram perturbations generated from the training data become the outcome of data analysis.
4.6.4.3 Data Validation

During the *data validation* process, a test data set is evaluated based on the outcome of data analysis. The test data consists of lists of an original sentence and its proposed correction. For each original sentence, a word-level perturbation process is performed and at the end of the process, a list of feasible word perturbations (a *perturbation list*) is created. The perturbation list is created based on the *n*-gram perturbations generated from the training data set during the data analysis process. An example of the list is shown in Figure 4.5. The list consists of a perturbation score, a perturbed sentence, and a perturbation process applied (*insertion/deletion/substitution/transposition*). The perturbed sentence refers to the original sentence which has been perturbed in order to become a possible correction for the original sentence. A perturbation score is calculated based on the statistical model of error correction that I proposed in Equation (4.34).

The perturbation list is then sorted in a descending order according to the perturbation scores. I then find out whether the correct perturbation features within the top 1, 2, 3 or 4 proposed perturbations. **Top 1** indicates for every evaluation round, the number of times the actual correction is ranked highest in the list of perturbations. **Top 2** indicates how many times the actual correction is in the top 2 in the list of perturbations. **Top 3** refers to how many times the actual correction is located in the top three of the list, and so forth.

Let me define a **correct perturbed sentence** as a perturbed sentence that is identified as the actual correction of an original sentence. For instance, in Figure 4.5, an exact perturbed sentence has the highest perturbation score. Therefore, a score of **Top 1** will be incremented by one. Else a score of **Top 2**, **Top 3**, or **Top 4** will be incremented if the exact perturbed sentence is placed in the second, third or fourth place in the perturbation list. After a final evaluation round for each error model, an average of each rank is computed.
4.7 Results and Discussion

During the evaluation of my proposed statistical error correction model, I focus on testing ungrammatical sentences which contain one error only. This means all test sentences are sentences which consist of one error only.

Firstly, I evaluate the statistical error correction model on the Comprehensive error model with the three types of training data: Type 1, Type 2, and Type 3. Table 4.5 outlines the number of trigram perturbations generated from the Comprehensive error model for the three types of training data. Next, an evaluation of the error correction model is performed on other error models with the Type 3 training data set only. Each evaluation result is then compared to the result from Comprehensive. This is to examine if there is a specific error model which works better than Comprehensive. A statistical test will be applied to examine any significant difference of the performance between two error models. Finally based on the performance results derived, I will suggest which error model, the statistical error correction model works well with.
Table 4.5: Total of trigram perturbations generated from the Comprehensive error model for each training data set

<table>
<thead>
<tr>
<th></th>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>2322</td>
<td>5401</td>
<td>6069</td>
</tr>
</tbody>
</table>

4.7.1 Result for Comprehensive error model

After the proposed statistical error correction model is evaluated on the Comprehensive error model, an average of percentage value for each top ranked training data type is computed. As discussed in §4.6.3.4, there are 3 types of training data. The results are outlined in Table 4.6. I also present them in a graph as illustrated in Figure 4.6. The y-axis of the graph represents the average of percentage value for ranking Top 1, Top 2, Top 3, and Top 4. The x-axis presents the cut off point of top ranked correct perturbed sentences in a perturbation list. Three lines with different patterns and shades represent each the type of training data set as labelled in the legend of the graph.

As shown in Figure 4.6, using the Type 1 training data set, on average about 48% of the time, the error correction model is able to provide correct perturbed sentences as Top 1 ranked. Meanwhile an average percentage for Top 2 ranked goes up to 62%. This shows about 62% of the evaluation time, the error correction model proposes correct answers in top 2 of a perturbation list. About 70% on average, the correct perturbed sentences are within Top 4 ranked.

Next, using the Type 2 training data set, its Top 1 percentage drops down about 10% as compared to the percentage of Top 1 in Type 1. Despite this, as compared to Type 1 again, the percentage of Top 2 is similar to Top 1 of Type 1 and the average percentages of Top 3 and Top 4 in Type 2 have increased about 5%.

Although the error correction model suggests smaller numbers for Top 1 in Type 2, the correct perturbed sentence is more often placed within the top 4 list. For instance, Figure 4.7 presents a sample of test results produced from a given test input sentence which is tested on a different training data set. In Figure 4.7(a), a correct perturbed sentence is on the top list for Type 1. However, when the input sentence is tested on
Table 4.6: An evaluation result for the Comprehensive error model

<table>
<thead>
<tr>
<th>Training set</th>
<th>Type 1</th>
<th>Type 2</th>
<th>Type 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>48.3%</td>
<td>37.7%</td>
<td>38.3%</td>
</tr>
<tr>
<td>Top 2</td>
<td>61.9%</td>
<td>61.2%</td>
<td>62.0%</td>
</tr>
<tr>
<td>Top 3</td>
<td>68.7%</td>
<td>71.8%</td>
<td>72.5%</td>
</tr>
<tr>
<td>Top 4</td>
<td>70.5%</td>
<td>74.3%</td>
<td>75.1%</td>
</tr>
</tbody>
</table>

Figure 4.6: Average performance for the Comprehensive error model for each training data set

Type 2 (see Figure 4.7(b)), the correct perturbed sentence goes down to the second place of the list.

Now let’s see how the performance of the error correction model is, when test data is run on Type 3 training data. Figure 4.6 shows that the average of all top ranked for Type 3, is higher than the average of all top ranked for Type 2. The number of trigram perturbations in Type 3 (see Table 4.5) is 11% more than Type 2. This shows that the n-gram perturbations added in Type 3 become a good source for the error correction model for proposing more correct perturbed sentences. As an overall performance, about 75% of the time on average, the error correction model provides correct perturbed sentences within the top 4 list. Next I would like to report more
Figure 4.7: Comparison results between Type 1 and Type 2 on a same test input sentence.

results from different types of error model.

4.7.2 Result for PN-label-backoff error model

The content of PN-label-backoff error model is similar to Comprehensive but all proper names are changed to a symbol <pn>. By using the unique symbol to represent all proper names, the problem of data sparsity is expected to be reduced. As such various word perturbations are reduced as well so that I hope that the performance will be better than Comprehensive.

In order to compare the performance results between PN-label-backoff and Comprehensive, I only tested PN-label-backoff on Type 3 training data. I’m not interested to see the performance among different types of training data set but I am rather interested to see the performance of the error model as compared to Comprehensive.

Table 4.7 outlines the evaluation results and Figure 4.8 illustrates graphically the
Table 4.7: The evaluation results for the *Comprehensive* and *PN-label-backoff* error models

<table>
<thead>
<tr>
<th>Error Model</th>
<th>Comprehensive</th>
<th>PN-label-backoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>38.3%</td>
<td>39.7%</td>
</tr>
<tr>
<td>Top 2</td>
<td>62.0%</td>
<td>62.4%</td>
</tr>
<tr>
<td>Top 3</td>
<td>72.5%</td>
<td>72.8%</td>
</tr>
<tr>
<td>Top 4</td>
<td>75.1%</td>
<td>75.3%</td>
</tr>
</tbody>
</table>

Figure 4.8: Comparison of average performance between *Comprehensive* and *PN-label-backoff* error models

performance between the two error models. Despite reducing the data sparsity problem, the performance result for *PN-label-backoff* is not better than *Comprehensive*. Even though all top ranked show increases the difference is not significant. What I can conclude here is that by reducing the problem of data sparsity, the performance is not improved. I carried out a hand inspection on the detailed results of test data. I noticed that the error correction model was able to propose correct perturbed sentences within the Top 4 ranked for test data which has the &lt;pn&gt; symbol.
4.7.2.1 Building a separate error model from PN-label-backoff error model

Based on the hand inspection performed on the detailed results of test data for PN-label-backoff, I performed another test. I narrowed down the scope of PN-label-backoff error model by only focusing on sentences which contain the <pn> symbol. For instance, suppose four sentence perturbations in (61) below. Therefore only (61a) and (61b) are considered.

(61) a. (“I from <pn>”, “I am from <pn>”)
    b. (“<pn>”, “<pn>”)
    c. (“He sells many fish at the market”, “He sells many fish at the market”)
    d. (“I to go school”, “I go to school”)

I named the newly created error model PN-sentences-label-backoff. From Comprehensive, I removed all sentences which do not contain any proper names and named the resulting sentences as PN-sentences. Therefore, PN-sentences and PN-sentences-label-backoff data are identical, except for the <pn> symbol. An evaluation was performed on both created error models. Table 4.8 outlines the average performance of PN-sentences and PN-sentence-label-backoff compared to Comprehensive and Figure 4.9 graphically shows the results.

In terms of performance results of PN-sentences-label-backoff as compared to PN-sentences, only Top 1 ranked average percentage increases about 4%. Although the increment is quite low, a paired two-tailed significance test (t-test) with 95% confidence level is applied. To prove there is a significant difference, the p-value must be below 0.05, (p < 0.05). The t test result is p = 0.007. It means there is a significant difference for Top 1 percentage between PN-sentences-label-backoff and PN-sentences as shown in Table 4.9. The result of the significance test shows that when the sparse data problem is reduced, the proposed statistical model of error correction is able to provide more correct perturbed sentences than on PN-sentences. However, other top ranked percentage values are not much different between the two error models.

I now compare top ranked average performance between PN-sentences-label-backoff and Comprehensive. The average percentage obtained by PN-sentences-
Table 4.8: The evaluation results for the Comprehensive, PN-sentences, and PN-sentences-label-backoff error models

<table>
<thead>
<tr>
<th>Error Model</th>
<th>Comprehensive</th>
<th>PN-sentences</th>
<th>PN-sentences-label-backoff</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>38.3%</td>
<td>70.2%</td>
<td>74.1%</td>
</tr>
<tr>
<td>Top 2</td>
<td>62.0%</td>
<td>76.9%</td>
<td>77.3%</td>
</tr>
<tr>
<td>Top 3</td>
<td>72.5%</td>
<td>78.2%</td>
<td>78.2%</td>
</tr>
<tr>
<td>Top 4</td>
<td>75.1%</td>
<td>78.6%</td>
<td>78.8%</td>
</tr>
</tbody>
</table>

Figure 4.9: Comparison of average performance among three error models

Table 4.9: Statistical tests for Top 1 performance between PN-sentences and PN-sentences-label-backoff

Two-tailed paired $t$ test with 95% confidence level

$paired-t(PN-sentences, PN-sentences-label-backoff) = -5.1017, p = 0.007$
Table 4.10: Statistical tests for Top 1 and Top 2 performance between PN-sentences-label-backoff and Comprehensive error models

<table>
<thead>
<tr>
<th>Ranked</th>
<th>Two-tailed t test with 95% confidence level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>( t(\text{Comprehensive, PN-sentences-label-backoff}) = -6.8595, p = 0.01025 )</td>
</tr>
<tr>
<td>Top 2</td>
<td>( t(\text{Comprehensive, PN-sentences-label-backoff}) = -5.0592, p = 0.03444 )</td>
</tr>
</tbody>
</table>

label-backoff is higher than that obtained by Comprehensive, for the Top 1 cutoff, and also for the Top 2, Top 3 and Top 4 cutoffs. The largest difference in results is about 36%: this was obtained for the Top 1 cutoff. In order to find out whether the difference of values are significant, the two-tailed significance test is again applied. Table 4.10 shows the difference between PN-sentences-label-backoff and Comprehensive error models for Top 1 and Top 2 ranked only are statistically significant.

4.7.3 Result for Tense error model

Next, I ran an evaluation of the proposed error correction model on a more specific error model. Here, I categorise a learner corpus into tense forms. Three tense forms are identified here: Present-tense, Past-tense, and Future-tense.

4.7.3.1 Present tense

A performance result for Present-tense error model compared to Comprehensive is outlined in Table 4.13 and graphically illustrated in Figure 4.10. The result shows that the average percentage for all top ranked in Present-tense is higher than Comprehensive. The average percentage for Top 1 ranked over all training data types in Present-tense has increased about 31% compared to Comprehensive. In order to find out any significant difference for each top ranked between the two error models, the two-tailed significance test is applied. Table 4.12 shows there is a significant difference between Present-tense and Comprehensive for Top 1, Top 2 and Top 3 ranked.
Table 4.11: The evaluation results for the *Comprehensive* and *Present-Tense* error models

<table>
<thead>
<tr>
<th>Error Model</th>
<th>Comprehensive</th>
<th>Present-Tense</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>38.3%</td>
<td>69.2%</td>
</tr>
<tr>
<td>Top 2</td>
<td>62.0%</td>
<td>75.0%</td>
</tr>
<tr>
<td>Top 3</td>
<td>72.5%</td>
<td>76.6%</td>
</tr>
<tr>
<td>Top 4</td>
<td>75.1%</td>
<td>78.4%</td>
</tr>
</tbody>
</table>

Figure 4.10: Comparison of average performance between *Present-tense* and *Comprehensive* error models

Table 4.12: Statistical tests for Top 1, Top 2, and Top 3 performance between *Present-tense* and *Comprehensive* error models

<table>
<thead>
<tr>
<th>Ranked</th>
<th>Two-tailed t test with 95% confidence level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>$t(\text{Comprehensive, Present-tense}) = -16.6626, \ p = 0.0000001$</td>
</tr>
<tr>
<td>Top 2</td>
<td>$t(\text{Comprehensive, Present-tense}) = -8.9462, \ p = 0.000001$</td>
</tr>
<tr>
<td>Top 3</td>
<td>$t(\text{Comprehensive, Present-tense}) = -2.5067, \ p = 0.003995$</td>
</tr>
</tbody>
</table>
Table 4.13: The evaluation results for the Comprehensive and all Tense error models

<table>
<thead>
<tr>
<th>Error Model</th>
<th>Comprehensive</th>
<th>Present-Tense</th>
<th>Past-Tense</th>
<th>Future-Tense</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>38.3%</td>
<td>69.2%</td>
<td>47.8%</td>
<td>42.5%</td>
</tr>
<tr>
<td>Top 2</td>
<td>62.0%</td>
<td>75.0%</td>
<td>63.2%</td>
<td>64.6%</td>
</tr>
<tr>
<td>Top 3</td>
<td>72.5%</td>
<td>76.6%</td>
<td>67.5%</td>
<td>71.5%</td>
</tr>
<tr>
<td>Top 4</td>
<td>75.1%</td>
<td>78.4%</td>
<td>69.5%</td>
<td>74.2%</td>
</tr>
</tbody>
</table>

4.7.3.2 Past-tense and Future-tense

Here, I explain the performance of the error correction model when it is evaluated on Past-tense and Future-tense error models. Table 4.13 outlines the performance results and Figure 4.11 depicts the comparison of performance results between all Tense sub-models and Comprehensive. Unfortunately, the result for Past-tense and Future-tense are not as good as Present-tense. Although Top 1 and Top 2 values are higher than Comprehensive, they are still much lower than Top 1 and Top 2 ranked in Present-tense. A significance test is applied for Top 1 percentage for Past-tense and Future-tense compared to Comprehensive. See Table 4.14. The test reveals that the percentage of Top 1 in Past-tense is significantly higher than Comprehensive but not in Future-tense. Other top ranked are also not significant. Yet, Top 3 and Top 4 percentage values for Past-tense and Future-tense drop down as compared to Comprehensive. Unlike Present-tense where all top ranked percentage values are higher than Comprehensive. As an overall for Tense sub-models, the proposed statistical error correction works better on the Present tense error model.

4.7.4 Result for Pronoun error model

As mentioned earlier in §4.6.3.3, the Pronoun error model is grouped into four sub-models: 1st person singular, 1st person plural, 3rd person singular and 3rd person plural. Refer back to Table 4.3 to see the example of pronouns. Table 4.15
Figure 4.11: Comparison of average performance between Tense and Comprehensive error models

Table 4.14: Statistical tests for Top 1 performance for Past-tense and Future-tense compared to Comprehensive

<table>
<thead>
<tr>
<th>Two-tailed paired t test with 95% confidence level</th>
</tr>
</thead>
<tbody>
<tr>
<td>paired-t(Past-tense, Comprehensive) = -3.1541, p = 0.02593</td>
</tr>
</tbody>
</table>

outlines and Figure 4.12 shows graphically all results for Pronoun sub-models as well as the Comprehensive error model. Among the four sub-models, only 3rd person singular performs better than Comprehensive. Table 4.16 shows Top 1 and Top 2 percentage values are statistically significant different from Comprehensive.

As for 1st person singular, Top 1 and Top 2 are lower than Comprehensive but Top 3 and Top 4 are quite similar to Comprehensive. The worst is 1st person plural where all top ranked are lower than top ranked in Comprehensive. What I can conclude here is that among the four error models as compared to Comprehensive, the proposed statistical error correction works best on the 3rd person singular error model.
Table 4.15: The evaluation results for the *Comprehensive* and *Pronoun* error models

<table>
<thead>
<tr>
<th>Error Model</th>
<th>Comprehensive</th>
<th>1st plural</th>
<th>1st singular</th>
<th>3rd plural</th>
<th>3rd singular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>38.3%</td>
<td>15.8%</td>
<td>28.1%</td>
<td>41.4%</td>
<td>56.8%</td>
</tr>
<tr>
<td>Top 2</td>
<td>62.0%</td>
<td>47.9%</td>
<td>58.3%</td>
<td>55.7%</td>
<td>68.7%</td>
</tr>
<tr>
<td>Top 3</td>
<td>72.5%</td>
<td>58.1%</td>
<td>71.7%</td>
<td>61.8%</td>
<td>73.2%</td>
</tr>
<tr>
<td>Top 4</td>
<td>75.1%</td>
<td>56.8%</td>
<td>68.7%</td>
<td>73.2%</td>
<td>75.6%</td>
</tr>
</tbody>
</table>

Figure 4.12: Comparison of average performance between *Pronoun* and *Comprehensive* error models

Table 4.16: Statistical tests for Top 1 and Top 2 average performance between 3rd person singular and *Comprehensive* error models

<table>
<thead>
<tr>
<th>Ranked</th>
<th>Two-tailed t test with 95% confidence level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>( t(\text{Comprehensive, 3rd person singular}) = -7.2085, \ p = 0.0006 )</td>
</tr>
<tr>
<td>Top 2</td>
<td>( t(\text{Comprehensive, 3rd person singular}) = -2.759, \ p = 0.003 )</td>
</tr>
</tbody>
</table>

4.7.5 Result for Question Types error model

Finally, I separate *Comprehensive* in two groups. The first group represents sentences or responses from Wh questions (\textit{Wh-q}) and the second group represents re-
responses from open-ended questions (Open-q). Table 4.4 on page 173 lists the sample of questions for each group. Table 4.17 outlines and Figure 4.13 graphically illustrates the evaluation results. Results show that the performance of Wh-q is comparable to Comprehensive. This means there is not much difference in terms of all top ranked values between both error models. However, in Open-q, the percentage of Top 1 ranked jumps more than 10% compared to Comprehensive. A statistical test revels there is a significant different for Top 1 percentage in Open-q. Unfortunately, other top ranked values drop drastically. As for which error model work better, I choose Open-q over Wh-q because the proposed error correction model is able to provide more correct perturbed answers than in Wh-q.

4.7.6 Discussion

Table 4.18 outlines and Figure 4.14 depicts the performance results for Comprehensive and the error model from each specific context group that performs the best. First of all, let’s view the figure in terms of patterns of how each rank improves within each error model. It clearly shows that the Comprehensive model improves faster than other sub-models, especially from Top 1 to Top 2 ranked, and from Top 2 to Top 3 ranked.

Now, I compare the performance of the different error models for the Top1, Top2, Top3 and Top 4 metrics individually. After comparing all performance results in the figure, my proposed statistical error correction model is able to provide more correct perturbed sentences specifically when it is evaluated on Present-tense and PN-sentence-label-backoff error models.

PN-sentences-label-backoff error model consists of sentences which contain proper names and the proper names are replaced by a <pn> symbol. This is to reduce the problem of sparse data because there are many different proper names. When an evaluation is performed on PN-sentences-label-backoff and PN-sentences as mentioned in §4.7.2.1, the proposed error correction model is able to provide more correct answers (Top 1) than when it is evaluated on PN-sentences-label-backoff. Even though the increment of Top 1 is not much it proved to be statistically significant.
Table 4.17: Results for Comprehensive, Wh-q, and Open-q error models

<table>
<thead>
<tr>
<th>Error Model</th>
<th>Comprehensive</th>
<th>Wh-q</th>
<th>Open-q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>38.3%</td>
<td>38.5%</td>
<td>50.3%</td>
</tr>
<tr>
<td>Top 2</td>
<td>62.0%</td>
<td>62.1%</td>
<td>58.8%</td>
</tr>
<tr>
<td>Top 3</td>
<td>72.5%</td>
<td>72.4%</td>
<td>61.6%</td>
</tr>
<tr>
<td>Top 4</td>
<td>75.1%</td>
<td>76.1%</td>
<td>63.8%</td>
</tr>
</tbody>
</table>

Table 4.18: The evaluation results for the Comprehensive and each specific context group that performs the best from all error models

<table>
<thead>
<tr>
<th>Error Model</th>
<th>Comprehensive</th>
<th>Open-q</th>
<th>Present tense</th>
<th>PN-sentences-label-backoff</th>
<th>3rd singular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 1</td>
<td>38.3%</td>
<td>50.3%</td>
<td>69.2%</td>
<td>74.1%</td>
<td>56.8%</td>
</tr>
<tr>
<td>Top 2</td>
<td>62.0%</td>
<td>58.9%</td>
<td>75.0%</td>
<td>77.3%</td>
<td>68.7%</td>
</tr>
<tr>
<td>Top 3</td>
<td>72.5%</td>
<td>61.6%</td>
<td>76.6%</td>
<td>78.2%</td>
<td>73.2%</td>
</tr>
<tr>
<td>Top 4</td>
<td>75.1%</td>
<td>63.8%</td>
<td>78.4%</td>
<td>78.8%</td>
<td>75.6%</td>
</tr>
</tbody>
</table>

Present-tense error model consists of sentences which are responses to questions in Present Tense form. In comparison to Comprehensive, the percentage of Top 1, Top 2 and Top 3 ranked are significantly improved. Why does the proposed error correction model work better with these two error models? Why not the other error models? I argue that it is a consequence of content and how many of n-gram perturbations generated from both error models help my proposed statistical error correction model in suggesting more correct answers (or correct perturbed sentences). In order to get more correct answers for erroneous sentences, we need training data that has a history of n-gram perturbations to represent the error correction. My proposed model suggests a list of corrections based on available n-gram perturbations in the training
Figure 4.13: Comparison of average performance between *Question Types* and *Comprehensive* error models.

Figure 4.14: Comparison of average performance among each specific context group that performs the best from all error models.
data. The highest chances for the correction to become top of the list is when a number of $n$-gram perturbation for a certain correction occurs frequently in the training data or the corpus.

I outline the number of trigram perturbations generated for each error model for Type 3 training data in Table 4.19. The actual total of $n$-gram perturbations is four times the count because for each trigram perturbation, two bigram and one unigram perturbations are generated. Among the Tense error model, the highest generated $n$-gram is Present-tense. Similarly to 3rd person singular case. My proposed error correction works best in both error models within their specific group. However, from Question-types, the lowest $n$-gram perturbations count is Open-q but its Top 1 ranked correct answer is higher than Wh-q. What I can conclude here is that the number of $n$-gram perturbations generated only does not affect the performance results. But the content of $n$-gram perturbations that help the error correction model in proposing more exact perturbed sentences is more important.

4.8 Summary

In this chapter, I have proposed a statistical model of error correction using the data gathered in the empirical study reported in Chapter 3. The model is developed based on language modelling technologies in which I applied the Witten-Bell discounting and the Backoff techniques. The model has been evaluated using $n$-fold cross-validation techniques on various error models and training data types. Results show that a convincing result has been achieved. The error correction model works best when the data sparsity problem is attended, i.e. when all proper names are replaced with a unique name or symbol. Examples are PN-sentence-label-backoff and more specific context such as Present-tense.

In the remaining chapters, the error correction model will be implemented on the Kaitito dialogue-based CALL system, and the error correction model will be tested in practice. The Kaitito system will be run online and language learners will be invited to have dialogue sessions with Kaitito. Results from the learners’ written conversations with Kaitito will be recorded. The conversations are then will be analysed and some
Table 4.19: Total number of trigram perturbations generated from each error model

<table>
<thead>
<tr>
<th>Error model</th>
<th>Trigram perturbations count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comprehensive</td>
<td>6069</td>
</tr>
<tr>
<td>PN-label-backoff</td>
<td>6041</td>
</tr>
<tr>
<td>PN-sentences-label-backoff</td>
<td>1219</td>
</tr>
<tr>
<td>PN-sentences-label</td>
<td>1262</td>
</tr>
<tr>
<td>Tense</td>
<td></td>
</tr>
<tr>
<td>Present-tense</td>
<td>2241</td>
</tr>
<tr>
<td>Past-tense</td>
<td>1929</td>
</tr>
<tr>
<td>Future-tense</td>
<td>1903</td>
</tr>
<tr>
<td>Pronoun</td>
<td></td>
</tr>
<tr>
<td>3rd person singular</td>
<td>1877</td>
</tr>
<tr>
<td>3rd person plural</td>
<td>767</td>
</tr>
<tr>
<td>1st person singular</td>
<td>2743</td>
</tr>
<tr>
<td>1st person plural</td>
<td>693</td>
</tr>
<tr>
<td>Question-types</td>
<td></td>
</tr>
<tr>
<td>Open-q</td>
<td>529</td>
</tr>
<tr>
<td>Wh-q</td>
<td>5543</td>
</tr>
</tbody>
</table>
conclusions will be made.
Chapter 5

Implementation of a Practical Dialogue-based CALL System

In this chapter, I will describe the implementation of a practical error correction system, designed to be used during a language learning dialogue, which incorporates the statistical model of error correction that I proposed in Chapter 4. The model is implemented within the Kaitito system, a dialogue-based CALL system. The system was designed to be used by the same group of learners and the same task domain for which data was gathered in Chapter 3, and for which the statistical model was configured.

The implementation involved a few stages. Firstly, I had to design a symbolic grammar for the Kaitito system to use, which was configured for the target group of learners, and the task they are given. I will discuss this process in §5.1.1 and §5.1.2.

Secondly, I had to design a simple dialogue manager for Kaitito, which asked the students a number of questions, analysed the student’s answers, and gave appropriate teaching responses. This dialogue manager will be discussed in §5.2 and §5.3. In §5.4, I show screenshots of Kaitito while executing the system. Lastly, I will talk about the programming code that I wrote in §5.5.
5.1 Building a Grammar for the Domain

I tried two methods for obtaining a grammar. The first was to use the full English Resource Grammar (ERG) which was described in §2.3.4. The second method was to use a reduced version of the ERG customised to my domain.

5.1.1 Parsing with the ERG

The reason I need a grammar is to decide whether the students’ sentences are well-formed or not. Kaitito needs to be able to parse the sentences which students type in, to determine whether they have grammatical errors or not. Recall that my error correction system cannot decide that itself: all it does is to suggest the best corrections for a sentence, assuming it contains an error. Luckily, the corpus of learner data that I gathered during the empirical study in Chapter 3 is a very rich source of information for evaluating candidate grammars to see if they can accurately distinguish between grammatical and ungrammatical sentences. The corpus consists not only of the learners’ actual utterances, but also judgements of native speakers about the well-formedness of the utterances. All sentences were annotated as grammatical or ungrammatical by English native speakers. Refer to Chapter 3 for further information on how annotation was carried out.

I created two lists from the learner corpus: a grammatical list and an ungrammatical list. My main goal is to see how good the ERG is at correctly parsing the grammatical sentences, and correctly rejecting the ungrammatical ones. However, I anticipated that the parser would have particular problems with proper names. For this reason, I created two versions of the grammatical list. The first one is the original grammatical sentences. I named this version Orig-crt-sentce. The content of the second version list is similar except in the second version, I converted all proper names to common English proper names which are known to be in the ERG lexicon. For example, all country names are changed to “New Zealand”, city names to “London”, and people’s names to “John”. I named the second version Orig-crt-same-PN. This group of sentences was altered to test the grammar of the ERG independently of its lexicon of proper names. The total number of sentences in Orig-crt-same-PN is less than the
number of sentences in **Orig-crt-sentce** because I removed all identical sentences from the former set.

As for *the ungrammatical sentences*, I paid special attention to sentences which have one error only. This is because my error correction model was built and tested on sentences which contain of one error only. I built a list of sentences with exactly one error which I called **One-err-sentce**. Table 5.1 outlines the three lists together with their total numbers of sentences.

I used the ERG (the version of July 2007) (Flickinger, 2000; Copestake and Flickinger, 2000) to parse the sentences in the lists I created. In order to avoid unknown word problems in the **Orig-crt-sentce** list, I only parsed sentences in **Orig-crt-same-PN** and **One-err-sentce** lists. Table 5.2 outlines the parsing results. Based on the parsing results, I categorised each sentence into one of the following four cases:

**Case 1: CorrectCorrect**: A sentence which is considered as “correct” by native speakers and the ERG.

**Case 2: ErrorError**: A sentence which is considered as “error” by the native speakers and the ERG.

**Case 3: CorrectError**: A sentence which is considered as “correct” by the native speakers but “error” by the ERG.

**Case 4: ErrorCorrect**: A sentence which is considered as “error” by the native speakers but “correct” by the ERG.

Table 5.2 shows 96% of the total sentences in **Orig-crt-same-PN** can be parsed (accepted by ERG) and only 4% are unparsed (unaccepted by ERG). Hence, there are 1235 from 1292 sentences are under the *CorrectCorrect* case. In the **One-err-sentce** list, 51% of the total sentences are the *ErrorError* case. About 49% of total ungrammatical sentences are marked as *CorrectError*. This shows that almost 50% of the total ungrammatical sentences in the **One-err-sentce** list can be parsed by the full ERG. In the following paragraph, I justify what situation occurs in the ERG.

The ERG used here has a broad coverage of English grammar. There are certain ungrammatical sentences which can be parsed using the full ERG. Suppose an answer
Table 5.1: Groups of sentences in the learner corpus

<table>
<thead>
<tr>
<th>List name</th>
<th>Description</th>
<th>Total of sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig-crt-sentce</td>
<td>All correct sentences that are annotated as correct sentences within a given context.</td>
<td>1822</td>
</tr>
<tr>
<td>Orig-crt-same-PN</td>
<td>All correct sentences that are annotated as correct sentences within a given context but all proper names are changed to common English names.</td>
<td>1292</td>
</tr>
<tr>
<td>One-err-sentce</td>
<td>All incorrect sentences which have one error only.</td>
<td>1337</td>
</tr>
</tbody>
</table>

Table 5.2: Parsing results for Orig-crt-same-PN and One-err-sentce using the ERG

<table>
<thead>
<tr>
<th>List name</th>
<th>Total ofParsed</th>
<th>Total of Unparsed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig-crt-same-PN</td>
<td>1235 (96%) (CorrectCorrect)</td>
<td>57 (4%) (CorrectError)</td>
</tr>
<tr>
<td>One-err-sentce</td>
<td>649 (49%) (ErrorCorrect)</td>
<td>688 (51%) (ErrorError)</td>
</tr>
</tbody>
</table>
for the question “Where is he from?” is

*He from Malaysia.*

Although the answer is accepted by ERG, its parse tree is graphically shown in Figure 5.1, the answer is grammatically incorrect by any sensible standards. The proper answer should be “*He is from Malaysia.*”, using the verb ‘*is*’. What has happened is the ERG has found a syntactically correct, but very unusual interpretation of the student’s sentence. This interpretation uses a rule in the ERG which allows any noun phrase to be a sentence by itself (which is required to parse one-word answers to questions, such as “Who arrived?” “John.”), plus a rule which allows any noun phrase to be modified by a PP.

The ERG finds many surprising interpretations. For instance, the sentence

*MMy father is a teacher.*

can be parsed in two ways according to the ERG. Figure 5.2 shows the two parse trees. The correct parse tree is of course (a). The parse tree (b) would be appropriate for the sentence if it were stressed as follows:

*MMy, Father is a teacher!*

The language learners have a smaller grammar knowledge and vocabularies than native speakers. In order to develop a system for learning English especially for beginners, a small or subset of English grammar is perhaps sufficient. Therefore, I developed a reduced version of the ERG customised to the learners’ language.
5.1.2 Generating a reduced version of ERG

The basic problem with the ERG is that it contains too many rules, especially for modelling the language of an EFL learner. I decided to create a version of the ERG which was customised to the kind of sentences produced by my target students.

A module called SERGE was developed in the Lisp programming language, by a colleague Peter Vlugter to generate a subset of the ERG. SERGE is an abbreviation of Subset ERG Extractor. The module takes as input a set of sentences, and returns as output a subset of the ERG which is just big enough to over all of the sentences.\footnote{SERGE has not yet been described in detail in a published paper. But the way it works is basically by identifying the HPSG rule types used in all the selected syntactic structures, and including all of these plus (recursively) the types of which they are instances, so that a coherent subset of the HPSG type hierarchy is extracted.} In fact it takes something a little more precise: for any syntactically ambiguous sentences, the user must specify which of the alternative analyses is the correct one. The module therefore requires as its input a list of sentences, together with a parse number for each ambiguous sentence identifying the correct analysis. The parse number is output from the LKB parser. For instance, consider the above sentence My father is a teacher. In Figure 5.2, the sentence has two parse trees when parsing with ERG. The parse tree (a) is referred to by the Linguistic Knowledge Builder (LKB) parser as parse number 1 and

![Figure 5.2: Two parse trees for a sentence “My father is a teacher.”](image-url)
Figure 5.3: A sample of correct sentences used in generating a reduced-ERG version

((("My name is John." 1)
 ("My name is Sara Nora John." 2)
 ("My father's name is John." 2)
 ("My best friend's name is John." 1)
 ("I am from London." 1)
 ("I live in London." 2)
 ("I went to the mall with my mom." 10)
 ("They like playing bowling." 3)
 ("I want to be a doctor when I grow up." 29)
 ("She will study with me." 1)
)

After generating the reduced version of the ERG which I called the reduced-ERG, I parsed all sentences in Orig-crt-same-PN and One-err-sentce with this new grammar. Parsing results are depicted in Table 5.3. When parsing sentences with a small coverage grammar, situations where fewer correct sentences can be parsed and more ungrammatical sentences can’t be parsed are expected. The results show that in Orig-crt-same-PN, only 87% actually correct sentences are acceptable to the reduced-ERG, compared to 96% of correct sentences when using the full ERG version. (Note that the number of correct sentences acceptable to the reduced-ERG is not 100%, because the reduced grammar is constructed from a small sample of training sentences, but is tested on a different, much larger, set of unseen sentences.) However, in One-err-sentce, the percentage of unparsable sentences increases to 64% from 51%. Since I want my system to detect as many errors as possible, I decided to carry on using the
Table 5.3: Comparison of parsing results between *Orig-crt-same-PN* and *One-err-sentce* using full ERG and reduced-ERG

<table>
<thead>
<tr>
<th>List name</th>
<th>ERG versions</th>
<th>Total ofParsed</th>
<th>Total of Unparseable</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Orig-crt-same-PN</em></td>
<td>Full ERG</td>
<td>1235 (96%)</td>
<td>57 (4%)</td>
</tr>
<tr>
<td></td>
<td>Reduced-ERG</td>
<td>1124 (87%)</td>
<td>168 (13%)</td>
</tr>
<tr>
<td><em>One-err-sentce</em></td>
<td>Full ERG</td>
<td>649 (49%)</td>
<td>688 (51%)</td>
</tr>
<tr>
<td></td>
<td>Reduced-ERG</td>
<td>476 (36%)</td>
<td>861 (64%)</td>
</tr>
</tbody>
</table>

reduced-ERG in Kaitito.

Table 5.4 outlines the percentage results for accuracy, precision and recall for the full ERG and the reduced-ERG versions. Although the precision of reduced-ERG is less than the full ERG version, the recall and accuracy of reduced-ERG is greater than full ERG. The reduced-ERG has a limited vocabulary list. This is because the vocabulary list consists only of words that are in the list of sentences I used as the input for SERGE. I used the vocabulary list which is available for the ERG, instead of using the vocabulary list generated for the reduced-ERG. However, proper names in the ERG are more biased towards a western culture. Therefore, I added a list of proper names which are common to Malaysian cultures. Malaysia has three main races: Malay, Chinese and Indian. Therefore I added proper names for the three races as well as the names of places. I obtained Malaysian proper names and places from the Internet. Table 5.5 lists the numbers of lexical items I added to the reduced-ERG lexicon. A complete list is given in Appendix F. Adding the lexicon was done automatically by invoking a Lisp function developed by van Schagen and Knott (2004). I edited the function so that it reads a list of words in a text file as input. For each word in the list, its corresponding lexicon definition is created. Then each definition is added into the ERG vocabulary list.

5.2 A Simple Dialogue Manager for Kaitito

*Kaitito* is a dialogue-based CALL system which is designed to teach the English and Māori languages (Vlugter *et al.*, 2004). Details about *Kaitito* can be found in §2.3.4.
Table 5.4: Results for accuracy, precision and recall of the reduced-ERG grammar

<table>
<thead>
<tr>
<th>ERG versions</th>
<th>Precision</th>
<th>Recall</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full ERG</td>
<td>92%</td>
<td>51%</td>
<td>73%</td>
</tr>
<tr>
<td>Reduced-ERG</td>
<td>84%</td>
<td>64%</td>
<td>76%</td>
</tr>
</tbody>
</table>

Table 5.5: The Total Numbers of lexical added in the reduced-ERG

<table>
<thead>
<tr>
<th>Lexical</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>People's names</td>
<td>804</td>
</tr>
<tr>
<td>Places names</td>
<td>349</td>
</tr>
<tr>
<td>Nouns</td>
<td>17</td>
</tr>
<tr>
<td>Adjectives</td>
<td>6</td>
</tr>
</tbody>
</table>

on page 19. In my thesis, I only focus on an error correction system for English sentences. Refer to §4.3 on page 145 for further information about the error correction model. Therefore the next step is to develop a simple version of Kaitito that implements my error correction model. Once the development is completed, an evaluation of the performance of the error correction model will be carried out; that will be explained later in Chapter 6.

5.2.1 The Basic Structure of A Dialogue Session

The implementation of the error correction system in Kaitito involved the development of a simple dialogue manager, to create a simple tutorial session where the system asks a series of questions, giving a student an opportunity to answer each one. The dialogue session is a question-answer session. The Kaitito system posts a question and the user responds to the question. Due to the limited coverage of my grammar, the user can’t pose a question back to the system. The questions asked by the system are a subset of the questions asked to students during the empirical study I carried out (see Chapter 3). From the 45 questions asked in the empirical study, I selected 21 questions as shown in Figure 5.4. Further details about the empirical study were discussed in Chapter 3.
The dialogue flow is shown in Figure 5.5 and Figure 5.6. Let me explain how the dialogue session works. Firstly, Kaitito poses a question and the user responds to it. The user’s response could be either

1. a response to the question,

2. a response to skip the posed question, or

3. a response to exit from the system.

If the user doesn’t want to answer the posed question, he/she can skip the question. Then Kaitito will pose the next question. The user also can exit from the system any time during the session. Whenever the user does answer a question, the answer is given to the teaching response module, so that the system can assess it and provide feedback. The teaching response module is described below (5.3). The dialogue session ends after the user responds to the last question.
1. Reach the last question? 
   - yes: Say Goodbye 
   - no: System asks a question
     - Student answers the question
       - exit: 2 
       - skip-q: 3 
       - response: Start

Figure 5.5: A dialogue flowchart, continued next page
Invoke the Teaching Response Module to suggest candidate corrections

Praise to the learner

Move to a new question

The Teaching Response Module

Found?

yes

Provide teaching hints

no

Provide a default response

Update try#

try# ≥ 3?

yes

Move to a new question

no

Repeat the current question

Figure 5.6: Continuation from Figure 5.5
5.3 The Teaching Response Module

If a user’s answer to a question can be parsed, the system responds by giving some form of praise to the user. This is has its own problems. Just because the user’s answer can be parsed does not mean it is a good answer. It may be completely irrelevant to the question being asked. The full Kaitito system derives semantic representations of users’ utterances, and can determine whether they actually answer the question being asked, giving appropriate teaching responses if they do not. But such semantic errors are not the focus of my project. Refer to the complete Kaitito system in Vlugter et al. (2004); Lurcock (2005); Slabbers (2005), which can determine whether a grammatical correct sentence actually answers the question which was asked. If a response from the user can’t be parsed, the response is passed to the perturbation module. This is where most of the effort of my project is focussed.

5.3.1 The Perturbation Module

How the perturbation module works was discussed in detail in §4.4.1. The output of the module is a list of candidate perturbations. The list is sorted in descending order according to the score of perturbations. The top three perturbations are retained.

5.3.2 Parsing Candidate Perturbations

Note that the candidate perturbations returned by the perturbation module are not guaranteed to be grammatically well formed. It is important that the system does not propose a correction unless there is some evidence that it is at least a well-formed English sentence. To this purpose, each member of the list of candidate perturbations is itself given to Kaitito’s parser. Any candidate perturbations which fail to parse are removed from the list. The system then creates a correction suggestion based on each remaining candidate perturbation, and delivers the set of suggestions to the student in its response, offering the student another chance to answer the question. The user is given three chances to correct her/his response. If the third response still fails to parse, Kaitito proceeds to the next question.
To take an example of the whole process: Suppose a user’s response to a question “Where were you born?” is “I born in Singapore.”. The response is unparsable with the reduced-ERG, so it is given to the perturbation module. Suppose the outcome of the module is a list of candidate perturbations as presented below:

\[(62)\]

a. (“i born in singapore”, “i was born in singapore”, word-insertion, 0.97)
b. (“i born in singapore”, “i born am in singapore”, word-insertion, 0.45)
c. (“i born in singapore”, “i born in the”, word-insertion, 0.21)
d. (“i born in singapore”, “my born in singapore”, word-substitution, 0.2)
e. (“i born in singapore”, “born in singapore”, word-deletion, 0.17)
f. (“i born in singapore”, “i am born in singapore”, word-insertion, 0.17)
g. (“i born in singapore”, “i come born in singapore”, word-insertion, 0.11)

In these perturbation structures, (62), the first element represents the student’s sentence, the second is its perturbation sentence, the third is the perturbation action and the last element is the perturbation score. There are 3 types of perturbation action considered: word-insertion, word-deletion, and word-substitution, as discussed in §4.4.1.

Note that perturbations (62b), (62c), (62d), (62e), and (62g) are syntactically ill-formed. When the candidate perturbations in (62) are reparsed, and the unparsable candidate are removed from the list, the new list of candidate perturbations is as follows:

\[(63)\]

a. (“i born in singapore”, “i was born in singapore”, word-insertion, 0.97)
b. (“i born in singapore”, “i am born in singapore”, word-insertion, 0.17)

5.3.3 The Form of Correction Suggestions: Teaching Hints

Lyster and Ranta (1997) discuss how teachers use different ways of responding to students’ ill-formed sentences, in a normal language learning classroom. The responses are known as corrective feedback (CF) or teaching hints. Recall from §2.6 that a piece of CF is a response from an addressee to a speaker, where the addressee’s intention

\(^2\)I put a threshold value as 0.1 for perturbation score which means any perturbations which less than the value are ignored.
is to correct the speaker’s erroneous utterance. The provision of CF is beneficial to students during language learning, as agreed by Long (1996); Swain (2005). Refer to §2.6 for further details about corrective feedback. In the following section, I will use the terms CF and teaching hints interchangeably.

When designing an error correction system, it is important that it provides educationally useful responses to students. Although each candidate perturbation consists of a complete sentence, I did not want to simply echo these complete sentences back to the users, for two reasons. Firstly, I want the user to have a chance to try answering the question again, rather than just copying a model answer. Secondly, my candidate perturbations are still not guaranteed to be grammatically well-formed. The ERG isn’t always right, as we have seen. I don’t want my error correction system to ever provide an ungrammatical sentence as a model answer.

The form of teaching hints are different depending on the type of perturbation actions. If the action is word-insertion, I use the following template:

“Perhaps you need to use the word ‘X’ in your sentence.”

If the action is word-deletion, I use the following template:

“Perhaps you need to leave out the word ‘X’ in your sentence.”

If the action is word-substitution, I use the following template:

“Perhaps you need to replace the word ‘X’ with ‘Y’ in your sentence.”

An example, as shown in (63), two candidate perturbations remain. Since both perturbation actions are word-insertion, the additional word in each perturbation sentence is “was” and “am” respectively. As such, the corrective feedback that I provide has the following form:

That’s not quite correct! Perhaps you need to
1: use the word ‘was’ in your sentence.
2: use the word ‘am’ in your sentence.
Please try again.
5.3.4 A Default Response

During the reparsing process, there are cases when all perturbation sentences are rejected by the reduced-ERG. When this situation occurs, a default response is provided for that question. The default response just presents a template for the answer which the student should provide. For instance,

Well, your answer may look like this: I was born in _____.

5.4 A Sample Session with Kaitito

Here I will show screenshots of the Kaitito system while it is in operation. Firstly, an evaluation starts with some brief information about the system as depicted in Figure 5.7. Then, when the word ‘evaluation’ or ‘here’ is clicked, a new page is issued. The page shows that a new dialogue identification (id) has been created. Refer to Figure 5.8. When the dialogue id is clicked, a dialogue page is displayed. Here, the system starts posing a question as shown in Figure 5.9. On this page, a user can click on the Skip button to skip the question being asked. Then the system will pose the next question. The user may exit at any time from the system by clicking on the Exit button.

If the user’s response is successfully parsed, the Kaitito system acknowledges the response as depicted in Figure 5.10. If the user’s response cannot be parsed then the perturbation module is invoked. If there are (acceptable) candidate perturbations, the system provides teaching hints, as shown in Figure 5.11. Suppose the user tries again responding to the same question and the new response is parsed correctly, the system praises the user, as shown in Figure 5.12. If there are no (acceptable) candidate perturbations, a default response is provided by the system, as shown in Figure 5.13. Finally, when the user responds to the last question, Kaitito provides a farewell response, as depicted in Figure 5.14.
Dialogue technology for language learning

Kaitito Dialogue system is a computer-assisted language learning (CALL) system. The system operates over the web, and allows a student to engage in a free-ranging dialogue with the system. The student can practise grammar, and conversational skills by having a free-ranging dialogue about everyday topics, just as one might with a human language tutor.

System Evaluation

You are invited to evaluate Kaitito Dialogue system. A conversation between the system and the student starts with the system poses some questions. The student types his/her response into a text box on the website. The system will respond either giving a praise if the response is grammatical or some hints for corrections if the response is ungrammatical. An evaluation of the system can be found [here](https://halizah.otago.ac.nz/kaitito/).

A new dialogue id is created for you. Click on the id number.

110523114046-22LO7MEMEFSDF

Figure 5.7: Welcome page

Figure 5.8: A new dialogue id is created page
Hello! My name is Kaitito. You can practice your English with me. I will ask you some questions. Please type your response in a text box and then press Enter. If you answer correctly, I will praise you. Otherwise, I will suggest some corrections. Please choose one and answer again. If you want to skip the question, click the button Skip. If you want to stop this conversation, click the button Exit below. Let's start our conversation.

Kaitito

TK> What is your name?

You> My name is Halizah Basiron

TK> That's interesting!

TK> Which city are you from?

Figure 5.9: Dialogue session page

Figure 5.10: Acknowledgement of a correct response
Figure 5.11: Teaching hint for an unparsed sentence

Figure 5.12: Praise given to a correct response

Figure 5.13: A sample ill-formed sentence, with a default response from the system
Hello! My name is Kaito. You can practise your English with me. I will ask you some questions. Please type your response in a text box and then press Enter. If you answer correctly, I will praise you. Otherwise, I will suggest some corrections. Please choose one and answer again. If you want to skip the question, click the button Skip. If you want to stop this conversation, click the button Exit below. Let's start our conversation.

Kaito

I see...
What is your best friend's ambition?
She wants to be a doctor.
Ok!
Thanks for the lovely chat with you. See you later! Click the button Exit to quit.

Figure 5.14: Exit response page
5.5 Programming Issues

The Kaitito system was developed using the Lisp programming language that incorporates the LKB parser and the ERG grammar. See Knott and Vlugter (2008); Vlugter and Knott (2006); Lurcock (2005); Slabbers and Knott (2005); van der Ham (2005); van Schagen and Knott (2004); Knott and Wright (2003). The system consists of 4 main modules:

1. a module which consists of Lisp source code of the Kaitito system,
2. a module which consists of HTML, Java and Perl source code to interface the system with the Internet,
3. the reduced-ERG, and
4. the LKB parser.

Most of my programming work was on (1) and (2). The development of reduced-ERG was explained earlier in §5.1.2. In the source codes of the system module (1), there are 51 Lisp program files. From the 51 files, I worked by adding programming code in 8 program files and I created two new files named read-csv.lisp and levenshtein.lisp. The read-csv.lisp file reads and analyses data from a training corpus (§4.6) and data captured from the real implementation (will be discussed §6.2.4). The levenshtein.lisp performs the Levenshtein distance metric to calculate the differences between two strings in the perturbation module.

As for (2), I created an introductory page (in html) as shown in Figure 5.7. Figure 5.8 to 5.14 are the output of Perl and Java scripts. A file named web-interface.lisp works as a communicator between the system source codes and the scripts. Table 5.6 summarises the percentage of lines of source code I contributed and its explanation of programming development performed respectively.

5.6 Summary

In this chapter, I have described the development of a reduced coverage version of the ERG and the development of a simple dialogue manager which implements my statisti-
<table>
<thead>
<tr>
<th>No</th>
<th>Filename</th>
<th>Percentage</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>backoff.lisp</td>
<td>41%</td>
<td>Calculating perturbation scores which followed the formula of my proposed error correction model. See §4.3.</td>
</tr>
<tr>
<td>2</td>
<td>dialogue.lisp</td>
<td>20%</td>
<td>Manipulating a dialogue data structure.</td>
</tr>
<tr>
<td>3</td>
<td>interpret.lisp</td>
<td>71%</td>
<td>Parsing students’ sentences and providing teaching hints (§5.3).</td>
</tr>
<tr>
<td>4</td>
<td>levenshtein.lisp</td>
<td>70%</td>
<td>Performing Levenshtein distance metric to calculate how many differences in two strings.</td>
</tr>
<tr>
<td>5</td>
<td>perturb.lisp</td>
<td>78%</td>
<td>Processing the perturbation module.</td>
</tr>
<tr>
<td>6</td>
<td>read-csv.lisp</td>
<td>100%</td>
<td>Analysing data from training corpus (§4.6) and the real implementation (§6.2.4).</td>
</tr>
<tr>
<td>7</td>
<td>structure.lisp</td>
<td>12%</td>
<td>A definition of data structure for a dialogue manager.</td>
</tr>
<tr>
<td>8</td>
<td>tty-interface.lisp</td>
<td>46%</td>
<td>The system’s interface.</td>
</tr>
<tr>
<td>9</td>
<td>turn.lisp</td>
<td>71%</td>
<td>Processing a dialogue turn.</td>
</tr>
<tr>
<td>10</td>
<td>web-interface.lisp</td>
<td>58%</td>
<td>Working as an interface between the dialogue manager and the Internet.</td>
</tr>
<tr>
<td>11</td>
<td>dlg</td>
<td>90%</td>
<td>The web interface of the system that is written in Perl script.</td>
</tr>
<tr>
<td>12</td>
<td>dlg.js</td>
<td>90%</td>
<td>A program to control input with a scroll bar in a dialogue textbox (written in Java script).</td>
</tr>
</tbody>
</table>
cal model of error correction (a teaching system). I also summarised the programming I did and which program files I worked on during the whole process starting from an evaluation of my error correction model on a training corpus until the development of the dialogue manager for Kaitito. In the next chapter, I will describe the evaluation of my teaching system. During the evaluation, language learners will be invited to have a dialogue session with the system.
Chapter 6

Evaluation of the Error Correction Model

After my teaching system was implemented, I ran an evaluation. Learners of English as a foreign language (EFL) were invited to access the system. The main objective of the evaluation was to examine the performance of my statistical error correction model in providing appropriate corrections for ill-formed sentences. I performed an analysis on recorded interactions between the system and the learners.

In the following, I will start with the method of evaluation used (§6.1). Then in §6.2, I present results of the analysis. The results are discussed in §6.3.

6.1 Methods

6.1.1 Participants

Participants involved in the evaluation were secondary school pupils in Malaysia. They were pupils from Form 2 classes. They were 14 years old on average. They have had up to seven years of learning English at school. The school I chose is the same school\(^1\) as one of the schools where the data gathering study reported in Chapter 3 was conducted, and the age of group of the students was the same as for that study.

\(^1\)The selected school is *Sch1* as explained in §3.2.
6.1.2 Procedures

After obtaining the relevant ethical consents from New Zealand and Malaysia, I approached the school principal to seek permission to carry out an evaluation at the school. The principal agreed to participate in this evaluation. Then, the principal introduced me to the coordinator of English teaching. I explained to the coordinator the purpose of the evaluation and also the requirement of the evaluation to be carried out in a computer laboratory. The computers had to be equipped with an Internet connection because the system runs online. She suggested that I run the evaluation during Information and Communication Technology Literature (ICTL) lessons. This is because students have to go to a computer laboratory for this subject. The duration of ICTL is 1 hour 20 minutes. I chose four Form 2 classes and informed their teachers about the evaluation.

Before each of the evaluation sessions commenced, I conducted a briefing session with the participating students. In the session, I explained the purpose of the evaluation and what the students had to do during the evaluation.

6.1.3 Materials and System Usage

An evaluation started when a student accessed the Kaitito system. Interactions between the system and the student take the form of a dialogue, comprising questions asked by the system, answers by the student and teaching responses by the system. The dialogue is initiated by an introductory page about the system and the creation of a unique dialogue identification (id), and followed by a dialogue session of around 30 minutes’ duration. Kaitito asks a set of questions and the student responds to these questions. The question are a subset of those asked to students during the data gathering study. Refer to Figure 5.4 on page 204 of §5.2.1 to see the list of questions. They are about the student, her/his parents, and her/his best friends. They focus on the use of verb tenses, 1st and 3rd singular pronouns and 3rd plural pronouns. If the students do not want to carry on the conversation, they may exit any time from the session.

The participating school had two computer laboratories. I ran four evaluation sessions at four different times on different students. Initially, I was allowed to use one
laboratory only where the ICTL subject was conducted. In the first session, among 20 computers available in the laboratory, only 8 computers were running well and connected to the Internet. Therefore, students had to take turns during the first evaluation.

In the second evaluation session, I requested to use another computer laboratory after determining it was unoccupied. However, in the second computer laboratory, only 4 computers were working and connected to the Internet. A teaching colleague helped me during the supervision of the second evaluation session.

During the evaluations which were carried out, I faced some difficulties. Insufficient number of computers that were connected to the Internet, slow Internet connection, and a bug which caused the system to have problems handling multiple accesses, have all affected the evaluation of the system. Due to a slow Internet connection, responses from the system often arrived late. While waiting for the system’s response, some students tried to provide a second answer to the question they were posed. This led to the flow of dialogue between the system and the students becoming disordered. Multiple accesses to the system at the same time also created some problems. Some students were asked a given question more than once. I asked these students to exit from the system and start a new dialogue.

To minimise the problem of multiple accesses, I decided to conduct the third and fourth evaluation sessions in one computer laboratory only. However, during both these sessions, the remote server that was running Kaitito became unresponsive. The server was located in Department of Computer Science, University of Otago, New Zealand. I called off this evaluation. A similar problem happened with the last evaluation on the next day. To avoid the students waiting, I stopped the evaluation.

The system’s hang occurred because of a run-time error in the system in each session. The error occurred because of an unparsable student’s answer which caused the parser to run out of memory. The sentences causing the hangs were “i like to do some interesting activities like reading and playing computer games.” and “i like to have some interesting activities like reading and playing computer games.”. Unfortunately, I didn’t implement a method for recovering from such hangs automatically. However, the problems which occurred during the evaluation will be taken into account in my
future work.

6.1.4 Data Captured during the Evaluation

When a student started a dialogue with Kaitito, a unique dialogue id was created. The dialogue id provides information on how many separate dialogue sessions the system participated in. During each dialogue session, all sentences keyed in by the students were recorded, as well as all actions of the system. For each dialogue session, I recorded two types of data: a dialogue script between the student and Kaitito and a tabular transcript. The dialogue script is a flow of dialogue between the student and the system. Due to problems I faced during the evaluation, the global structure of dialogue scripts is often not well recorded. Some students’ responses are not associated with the right system questions. The same goes for the system’s default responses. Fortunately, the system’s responses are always correctly associated with the ill-formed student sentences which they are responding to. An example of a disorganised transcript is shown in Figure 6.1.

A few weeks after the evaluation was carried out, I contacted the ICTL subject teacher to ask her students to use Kaitito. I requested that only one student accessed the system at a time. The students who accessed Kaitito were the students whose evaluation sessions were called off due to the system’s run-time error. I obtained six new dialogues this way. This time all six transcripts are nicely ordered. Note that these transcripts are all arrived too late to be included in the analysis of the evaluation. A sample of a well-ordered dialogue transcripts is shown in Figure 6.2.

The second type of data I captured is a tabular transcript. The transcript consists of a list of data in a tabular form. An example of a tabular transcript is shown in Figure 6.3. Each data line in the tabular transcript consists of 4 elements: a question number, a student’s answer, status of the answer, and the response from the system. As mentioned earlier regarding the problems I faced during the evaluation, the global structure of tabular transcripts is often not well recorded. Some students’ answers are not associated with the right system questions. The same goes for the system’s default responses. Fortunately, the system’s responses are always correctly associated with the
----Start Dialogue----
ID: 110331162559-34IL1UFKNTHN0
1) What is your name?
You> my name is mohd arif bin dinan
Wonderful!
2) Which city are you from?
You> my name is halifi
Good!
3) How old are you?
Well, your answer may look like this: I am ___ years old.
3) How old are you?
You> Malaysia
Great answer!
4) Where were you born?
You> Hospital
Yes, that's right!
5) How many brothers and sisters do you have?
You> I born at Hospital Besar Melaka
That's not quite correct! May be you could
1: use the word 'was' in your sentence.
2: use the word 'am' in your response. Please try again.
5) How many brothers and sisters do you have?
You> Hospital
That's not quite correct! May be you could
1: use the word 'was' in your sentence.
2: use the word 'am' in your response. Please try again.
5) How many brothers and sisters do you have?
You> Malaysia
Good!
6) What do you like doing?
7) What is your best friend's name?
You> fitri
Yes, that’s right!
8) How many brothers and sisters does your best friend have?
You> 3
That's right!
9) How old is your best friend?
You> hazman
Good!
10) What does your best friend like doing?
You> I
Well done!
11) What do your parents like doing?
You> one million
Great answer!
12) What do you and your friends do together?
You> fishing
Wonderful!
13) What is your father's job?
You> kil
Wonderful!
14) What did you do last weekend?
You> read book
That's not quite right! Perhaps you have to
1: use the word 'books' in your answer.
2: use the word 'a' in your sentence.
3: use the word 'the' in your sentence. Please try again.
14) What did you do last weekend?
You> masjid tanah , melaka
Wonderful!
15) What did your best friend do last weekend?
You> malacca

Figure 6.1: A disordered sample dialogue script
-----Start Dialogue-----
ID: 110428154720-GF6G34FF321A
1) What is your name?
You> my name is fatini
Ok!
2) Which city are you from?
You> malacca
That's interesting!
3) How old are you?
You> 2 years old
Ok!
4) Where were you born?
You> sabah
Ok!
5) How many brothers and sisters do you have?
You> 4 brothers 12 sisters
That's not quite right! May be you could use the word 'and' in your response. Please try again.
6) What do you like doing?
7) What is your best friend's name?
You> fatini
Ok!
8) How many brothers and sisters does your best friend have?
You> 3
I see...
9) How old is your best friend?
You> 20 years old
I see...
10) What does your best friend like doing?
You> dancing
I see...
11) What do your parents like doing?
You> cleaning house
That's not quite right! Perhaps you could
1: replace the word 'house' with 'houses' in your sentence.
2: use the word 'my' in your sentence.
3: use the word 'the' in your response. Please try again.
11) What do your parents like doing?
You> my parents like to cleaning the houses
That's not quite right! May be you have to
1: use the word 'listening' in your answer.
2: leave out the word 'to' in your sentence.
3: use the word 'do' in your sentence. Please try again.
11) What do your parents like doing?
--------End Dialogue--------

Figure 6.2: A well structured sample dialogue script
ill-formed student answers which they are responding to.

For instance, a tabular data line is

(64) (“1” “my name is fatini” “correct” “Ok!”)

In (64), the first element, “1” represents the question number, and “my name is fatini” is the student’s answer. The third element represents an assessment of the correction in the answer, provided by the parsing process. The status “correct” means the student’s answer is parsed successfully. The last element represents Kaitito’s response.

If a student’s answer can’t be parsed, a sample of the data recorded is represented in example (65) below:

(65) (“5” “4 brothers 12 sisters” (“4 brothers and 12 sisters” 0.25)) “That’s not quite right! May be you could use the word ‘and’ in your response. Please try again.”)

The third element in (65) which is (“4 brothers and 12 sisters” 0.25) is the candidate perturbation. The fourth element which is “That’s not quite right! May be you could use the word ‘and’ in your response. Please try again.” is the system’s response,
delivering teaching hints based on the perturbation list back to the student.

If a student’s response can’t be parsed and no “grammatical” candidate corrections (acceptable by reduced-ERG) are available, the status of the response represented in the third element of a tabular list is “no-perts”. A default response from Kaitito is provided for no-perts responses. A sample of data recorded is represented in the example (66) below:

(66) (“5” “2 brother 1 sister” “no-perts” “Hmm, your response may look like this: I have ____ brothers and ____ sisters. Please try again.”)

If the student skipped the posed question, say the question number 6, the data is captured as in example (67).

(67) (“6” “” “” “”)

6.2 Results

An analysis was conducted on students’ tabular transcripts. The analysis involves three stages roughly corresponding to the pipeline of processing in the error correction system described in Chapter 4: preliminary checking, grammar evaluation and an evaluation of the error correction model. The evaluation of the grammar corresponds to a development of the medium coverage grammar called reduced-ERG, as described in §5.1. The last step corresponds to an evaluation of the statistical of error correction model as discussed in §4.6.

6.2.1 Summary of Students’ Responses

Let me first describe some basic statistics about the students’ responses. After the evaluation was done, I identified how many dialogue ids had been created. Four classes of students participated in the evaluation, which means 4 evaluation sessions were held. Table 6.1 outlines the number of dialogue ids created for each class. The total number of unique ids is 92. As shown in Table 6.1, fewer ids were created during the last two sessions because of the run-time error which occurred in the system.
Table 6.1: Dialogue ids created per class

<table>
<thead>
<tr>
<th>Class</th>
<th>Id created</th>
</tr>
</thead>
<tbody>
<tr>
<td>2A</td>
<td>30</td>
</tr>
<tr>
<td>2D</td>
<td>42</td>
</tr>
<tr>
<td>2B</td>
<td>9</td>
</tr>
<tr>
<td>2KAA1</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 6.2: Total of student answers provided

<table>
<thead>
<tr>
<th>Parsed</th>
<th>Unparseable</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1022</td>
<td>386</td>
<td>1408</td>
</tr>
</tbody>
</table>

I used tabular transcripts captured to conduct an analysis. In order to calculate the total numbers of parsed and unparsable answers, I compiled all data from tabular transcripts and kept them in one file. Figure 6.4 shows a sample of compiled tabular transcripts. Then I developed a Lisp function which reads the file as input and automatically calculates the number of parsed and unparsable answers. The total number of answers provided by students is 1408, of which 1022 were parsed and 386 were unparsable, as summarised in Table 6.2.

6.2.2 Preliminary Checking

A major problem faced by any parsing system is how to handle unknown words. At present, Kaitito’s parser deals with an unknown word by trying to assume the word is a proper name (van Schagen and Knott, 2004) and adding the word to the lexicon if the sentence can be parsed on this assumption. However, this strategy results in some student errors being missed. For example, a student’s sentence to answer the question “What does your best friend like doing?” is

(68) John likes drowing.

This response has an unknown word, drow ing. The sentence (68) is parsed successfully by Kaitito, and its parse tree is shown in Figure 6.5(a). This sentence is interpreted as John likes someone whose name is Drowing. However, according to the question
Figure 6.4: A sample compiled tabular transcript
Figure 6.5: A parse tree for a sentence with a misspelled word ‘drowing’ (a) and after the spelling is corrected (b)

The response must be saying that someone likes to do something. As such, the word *drowing* can be seen as the wrongly spelled word for *drawing*. Therefore the right sentence would be:

(69) *John likes drawing.*

Figure 6.5(b) is the parse tree of sentence (69) when it was parsed by Kaitito.

This case led me to perform a preliminary checking on each student’s response. The checking is to overcome the parser’s lexical shortcomings. I performed five types of correction by hand:

1. correcting spelling errors for English words,
2. converting short form words to the corresponding full form,
3. removing unnecessary symbols or blanks occurring in students’ answers,
4. separating two words that have symbols in between them without blanks, and
5. translating students’ words in L1\(^2\) to corresponding English words.

Table 6.3 shows a few examples of each limitation, together with its corresponding amendment.

---

\(^2\)L1 here is the Malay language.
<table>
<thead>
<tr>
<th>Case</th>
<th>Original Responses</th>
<th>Corresponding correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spelling errors</td>
<td>My father is a driver.</td>
<td>My father is a driver.</td>
</tr>
<tr>
<td>Spelling errors</td>
<td>I like to drawing and painting</td>
<td>I like to drawing and painting</td>
</tr>
<tr>
<td>Words separration</td>
<td>My name is ain, fatah, yohannah and fatini.</td>
<td>My name is ain, fatah, yohannah and fatini.</td>
</tr>
<tr>
<td>Unnecessary symbols or blanks</td>
<td>Ha // Badan</td>
<td>Ha</td>
</tr>
<tr>
<td>Short form words</td>
<td>I'll be going to the library</td>
<td>I'll be going to the library</td>
</tr>
<tr>
<td>Missing words</td>
<td>Helping &amp; hardworking</td>
<td>Helping &amp; hardworking</td>
</tr>
<tr>
<td></td>
<td>Online, you?</td>
<td>Online, you?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6.4: Distribution of percentage on each preliminary checking case

<table>
<thead>
<tr>
<th>Case</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spelling errors</td>
<td>54%</td>
</tr>
<tr>
<td>Short form words</td>
<td>13%</td>
</tr>
<tr>
<td>Words separation</td>
<td>21%</td>
</tr>
<tr>
<td>L1 words</td>
<td>4%</td>
</tr>
<tr>
<td>Multiple sentences</td>
<td>4%</td>
</tr>
</tbody>
</table>

Table 6.5: Total of responses after preliminary checking done

<table>
<thead>
<tr>
<th>Parsed</th>
<th>Unparseable</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1052</td>
<td>361</td>
<td>1413</td>
</tr>
</tbody>
</table>

Another shortcoming of Kaitito’s parser is that it is unable to parse more than one sentence at one time. For example, a student’s answer which results in a parse failure is:

(70) My father. His like to reading newspaper

In cases like this, I parsed the individual sentences to the parser one by one, and recorded its teaching response to each sentence. Since the multiple sentences are separated, the new total of responses grows somewhat, it is now 1413.

After the above corrections were done, I reparsed all the affected sentences. About 25% of the total 1408 responses required some manual alteration during the preliminary checking stage. A distribution of percentage for each problem is shown in Table 6.4. More than half the corrections performed during the preliminary checking were spelling error corrections.

The new total of responses is summarised in Table 6.5. Of 1413 sentences, 74% are accepted by Kaitito as a correct parse, leaving only 26% unparsed. Of course I must still determine whether the parser’s decisions are correct. This is done in the second stage of analysis.
Table 6.6: Results for the analysis of students’ answers

<table>
<thead>
<tr>
<th></th>
<th>Human</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Parser</td>
<td>Error</td>
<td>309</td>
<td>52</td>
</tr>
<tr>
<td></td>
<td>Correct</td>
<td>85</td>
<td>967</td>
</tr>
</tbody>
</table>

6.2.3 Evaluating Grammar

Kaitito used a medium-coverage grammar which I called the reduced-ERG grammar. The development of the grammar was discussed in §5.1. In order to measure the precision, recall and accuracy of this grammar on the students’ responses, I manually classified all answers given by the students as syntactically correct or incorrect, and compared their assessments to the parser’s assessment. For each sentence, there are now four possibilities.

1. ErrorError represents a sentence which is considered as “error” by the human judge and the parser. If we treat the parser as a system which is trying to identify ungrammatical sentences, this constitutes a true positive.

2. CorrectCorrect or true negative represents a sentence which is considered as “correct” by the human judge and the parser.

3. ErrorCorrect or false negative represents a sentence which is considered as “error” by the human but “correct” by the parser.

4. CorrectError or false positive represents a sentence which is considered as “correct” by the human but “error” by the parser.

Table 6.6 outlines the total number of sentences which fell into each category.

Given these data, I calculated the precision and recall of the grammar. The formulas for precision and recall are respectively defined as follows:

$$\text{Precision} = \frac{\sum (\text{Error Error})}{\sum (\text{Error Error}) + \sum (\text{Correct Error})} \quad \text{(6.1)}$$

$$\text{Recall} = \frac{\sum (\text{Error Error})}{\sum (\text{Error Error}) + \sum (\text{Error Correct})} \quad \text{(6.2)}$$
Then, I calculated the *accuracy* of the grammar by taking the sum of the total number of *Error* sentences and total number of *Correct* sentences, and dividing by the total number of sentences in all categories. The formula for accuracy is defined as:

\[
\text{Accuracy} = \frac{\sum (\text{Error}) + \sum (\text{Correct})}{\sum (\text{all sentences in each category})} \tag{6.3}
\]

\[
= \frac{309 + 967}{309 + 52 + 85 + 967} = 0.90
\tag{6.4}
\]

The percentage results for accuracy, precision and recall are outlined in Table 6.7. The results show that the *reduced-ERG* grammar is capable of identifying sentences which contain errors at 86% precision, 78% recall and 90% accuracy. The percentage of *Correct* and *Error* is respectively 68% and 21%. These results are very competitive compared to Baldwin, Bender, Flickinger, Kim, and Oepen (2004)’s work. Baldwin *et al.* tested the ERG (July 2003 version) on a random sample of the British National Corpus. The sample consisted of 20,000 words of written text. The results were 57% correct parse and 43% parse failure. Our results show that a customised version of the ERG can be quite effective in modelling the language of EFL learners.

Comparing Table 6.7 and Table 5.4 (for the *reduced-ERG* grammar), the results in Table 6.7 are better than in Table 5.4. I am not sure why the results of this real evaluation are better than the evaluation of learner corpus in §5.1. But I am pleased with the results. Perhaps the students were making less mistakes because they maybe were being conservative in their language, because they were interacting with a computer. This would be explored in my future work.

Table 6.7: Results for accuracy, precision and recall of the *reduced-ERG* grammar

<table>
<thead>
<tr>
<th>Precision</th>
<th>Recall</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>86%</td>
<td>78%</td>
<td>90%</td>
</tr>
</tbody>
</table>
6.2.4 Evaluating The Error Correction System

In this evaluation stage, I focused on investigating the performance of the error correction model implemented in Kaitito. I wanted to know how well the error correction model provides candidate corrections for ill-formed sentences. Here, I only concentrated on students’ answers in the Error category, in which the total of answers is 309 (refer to the highlighted cell in Table 6.6).

As discussed in §5.3, when a response given by a student can’t be parsed, the perturbation module is invoked. The output of the module is a list of candidate perturbations which the system presents in the form of teaching hints. From the total of Error, 65% of 309 sentences are provided with candidate perturbations, (“with-perts”). The remaining 35% of sentences don’t have any acceptable (by reduced-ERG) candidate perturbations (“no-perts”). Figure 6.6 graphically shows the percentage distribution between with-perts and no-perts.

Firstly, I will evaluate the perturbation algorithm implemented in the error correction system (§6.2.4.1). Then, I will observe how feedback from the system response to students’ ill-formed sentences can be beneficial to students during language learning (§6.2.4.2). In the following I will use the terms candidates perturbations and candidate corrections interchangeably.
6.2.4.1 Evaluating the perturbation algorithm

This evaluation is to calculate for each sentence, how many have candidate perturbations that are the first (Top 1) and within the first three (Top 3) in the list of candidate perturbations. This analysis was performed manually, as illustrated by the following examples. Suppose, the student’s input to the question “Which city are you from?” is

(71) *I from Masjid Tanah.*

and its candidate perturbations list (sorted in descending order according to its perturbation score respectively) is as follows

(72) a. *I am from Masjid Tanah.*
    b. *I come from Masjid Tanah.*
    c. *I was from Masjid Tanah.*

I identify the first candidate, (72a) is the most appropriate correction, so the sentence (71) is marked as having a Top 1 ranked candidate correction. Another example is

(73) *I like to cycling.*

and its candidate perturbations list is as follows

(74) a. *I like listening to cycling.*
    b. *I like cycling.*
    c. *I like to do cycling.*

I identify the second candidate, (74b) as the most appropriate correction, so the sentence (73) is marked as having Top 3 ranked candidate correction.

The results of this evaluation are outlined in Table 6.8. 38% of all Error sentences have corrections which are Top 1 ranked. Yet, the percentage of Top 3 ranked corrections increases just 3% from Top 1 ranked. This shows that most of the correct perturbations are in the ranking Top 1.
Table 6.8: Results of the performance for Error Error sentences

<table>
<thead>
<tr>
<th>Top 1</th>
<th>Top 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>38%</td>
<td>42%</td>
</tr>
</tbody>
</table>

Table 6.9: Results of the performance for one-err sentences

<table>
<thead>
<tr>
<th>Top 1</th>
<th>Top 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>59%</td>
<td>66%</td>
</tr>
</tbody>
</table>

**Evaluation for sentences which have one error only**  My error correction model is tailored to provide corrections for ill-formed sentences which consist of one error only. Therefore, among the 309 responses, about 64% (198) are sentences with one error only (*one-err*). The results are outlined in Table 6.9 after I calculated the number ranking **Top 1** and **Top 3**. The result shows almost 60% of the total of *one-err* sentences have a **Top 1** ranked candidate correction. The percentage of **Top 3** ranked corrections is 7% more than **Top 1** ranked.

As described in §4.6, I performed an evaluation of my error correction model using a training corpus. As such, I can make a comparison between the results of the evaluation using the training data (Eval 1) and the results of *one-err* (Eval 2). Table 6.10 shows the results of Eval 1. The results show that the **Top 1** ranked of Eval 2 is more than 20% higher than Eval 1 but the percentage of **Top 3** of Eval 2 is less 8% than Eval 1. During Eval 1, I didn’t reparse each candidate perturbations (as I did in Eval 2) to remove all candidate perturbations that are rejected by Kaitito’s parser. Otherwise, I would say both performances may be similar.

Table 6.10: Results of the performance of error correction model evaluated on a training corpus

<table>
<thead>
<tr>
<th>Top 1</th>
<th>Top 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>38%</td>
<td>73%</td>
</tr>
</tbody>
</table>
6.2.4.2 Evaluating the provision of teaching hints

As mentioned earlier, my error correction system is designed to provide appropriate candidate perturbations (if any) for one-err sentences only. However, candidate perturbations are still provided (if any) to sentences which have more than one error (more-err). For more-err sentences, their candidate perturbations always suggest a correction to one error in the sentence. Even though the perturbations are not completely grammatical, I observed some examples of such teaching hints (provided based on the candidate perturbations) which are still useful because the hints suggest to students how to make improvements to their ill-formed sentences.

Long (1996) and Swain (2005) have agreed that the provision of teaching hints is beneficial to students during language learning. If a student manages to locate and correct error/s, then a different set of candidate perturbations may be provided, or a praise from the system, if the altered sentence is parsed. As an example, a student’s response to the question “What is your name?” is

\[ \text{My name Fatini.} \]

which can’t be parsed by the Kaitito system. Therefore, the perturbation module is invoked. There is only one candidate perturbation is acceptable which is

(75) \( \text{my name is fatini.} \)

Based on the candidate perturbation, the system responds with

(76) \( \text{That’s not quite right! Perhaps you could use the word ‘is’ in your sentence.} \)
\( \text{Please try again.} \)

The teaching hints given in (76) will give a chance to the students to figure out what mistakes they have made. Suppose the student rephrased her/his sentence as \( \text{My name is Fatini.} \). This shows that the student has tried to follow the hint given in (76) by inserting the word ‘is’ in the second trial. When the response is parsed, it is parsed successfully and the system praises to the student.

Another example is referred from Figure 6.2. Considering the second last answer from the student to the question “What do your parents like doing?” is
which can’t be parsed by Kaitito. Therefore, a list of candidate perturbations generated from the perturbation module is:

(77)  
   a. cleaning houses.
   b. cleaning my house.
   c. cleaning the house

Then the teaching hints provided based on (77) are:

(78)  That’s not quite right! Perhaps you could
      1. replace the word ‘house’ with ‘houses’ in your sentence.
      2. use the word ‘my’ in your response.
      3. use the word ‘the’ in your response.
      Please try again.

Based on Figure 6.2, the student’s last response after the teaching hints (78) was:

(79)  my parents like to cleaning the house.

The student’s second input leads to different teaching hints as follows:

(80)  That’s not quite right! Perhaps you could
      1. use the word ‘listening’ in your sentence.
      2. leave out the word ‘to’ in your response.
      3. use the word ‘do’ in your response.
      Please try again.

Based on all transcripts recorded during the evaluation, it is hard to find examples that showed a student managing to successfully correct the more-err sentences after a series of teaching hints being provided. However, I am able to find 10 cases when students are able to correct their sentences based on the provided teaching hints. Sample student transcripts are attached in Appendix G.

Teaching hints provided for the more-err sentences  From 309 ErrorError sentences, 200 have candidate perturbations. Then, from 200, only 38 are more-err
sentences. I performed an observation on the 38 more-err sentences to calculate how many of their candidate perturbations are in ranking Top 1 and Top 3. Suppose the sentence is

(81)  he like play football

has a list of candidate perturbations as follows

(82)  a. he likes play football.
    b. he will like play football.
    c. i he like play football.

then its corresponding teaching hints are

(83)  That’s is not quite right! Perhaps you could
    1. replace the word ‘like’ with ‘likes’ in your response.
    2. use the word ‘will’ in your sentence.
    3. use the word ‘i’ in your response.
    Please try again.

The candidate (82a) is considered as Top 1 ranked as I identified it as the proper correction for the sentence (81). The correction is to fix a subject-verb rule agreement error.

Results for the evaluation of the provision of teaching hints are outlined in Table 6.11. The table also shows the different scopes of ErrorError. The scope of with-perts (in the second row) is a subset of ErrorError. The more-err sentence (in the third row) is a subset of with-perts. When analysing all 309 ErrorError sentences, 44% of them have useful teaching hints at the ranking Top 1 and 49% at the ranking Top 3. However the performance for Top 1 and Top 3 shows improvement when I focus only on sentences that have candidate perturbations, as shown at the second row. From 200 sentences, 68% have useful teaching hints at the ranking Top 1 and 75% at the ranking Top 3. I narrowed down my analysis on the more-err sentences. Half of 38 sentences (50%) have corrections which lead to providing effective teaching hints at ranking Top 1 and Top 3. Since the percentage of both rankings is similar, this indicates that almost all candidate perturbations in 19 more-err sentences are Top 1 ranked.
Table 6.11: Results of the performance for useful teaching hints

<table>
<thead>
<tr>
<th>No</th>
<th>Scope of Sentences</th>
<th>Total sentences</th>
<th>Top 1</th>
<th>Top 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Error</td>
<td>309</td>
<td>44%</td>
<td>49%</td>
</tr>
<tr>
<td>2</td>
<td>A subset of Error with candidate perturbations (with-perts)</td>
<td>200</td>
<td>68%</td>
<td>75%</td>
</tr>
<tr>
<td>3</td>
<td>More-err: a subset of 2</td>
<td>38</td>
<td>50%</td>
<td>50%</td>
</tr>
</tbody>
</table>

### 6.3 Discussion

Here I would like to discuss the results that I gained after the analysis of students’ transcripts. Firstly, about the reduced-ERG grammar. The grammar showed a competitive performance in rejecting ill-formed sentences and accepting well-formed sentence with precision/recall of 86%/78%. The results are better than Lee (2009)’s and Chodorow et al. (2007)’s system. The best precision/recall result in Lee (2009) is 83%/73%, with 80%/30% in Chodorow et al. (2007). The accuracy of the grammar is 90%, which is far better (10% more) than Gamon et al. (2008) and Sun et al. (2007). Of course, it is slightly unfair to compare different systems quantitatively, since they are tailored to different users and different domains. But in the end, what is important for any teaching system is that its grammar has good coverage of the sentences in its domain, so it is not unreasonable to make direct comparisons of this kind.

Second is the evaluation of my statistical error correction system. Two analyses were done: of the perturbation algorithm, and of the provision of teaching hints. Table 6.12 summarises the overall results of the evaluation of perturbation algorithm. The results show from the total of 309, only 38% have corrections in the ranking Top 1 and 42% (or 131 sentences) have corrections in the ranking Top 3. The difference of both rankings is small, only 4%. This indicates that most of candidate perturbations in the 42% Top 3 are Top 1 ranked.

The statistical error correction system is tailored to provide (grammatical) corrections for ill-formed sentences which are one-err sentences only. Table 6.12 shows the comparison of results for Top 1 and Top 3 between the ErrorError and the one-err
<table>
<thead>
<tr>
<th>Types</th>
<th>% of Top 1</th>
<th>% of Top 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ErrorError</td>
<td>38%</td>
<td>42%</td>
</tr>
<tr>
<td>One-err</td>
<td>59%</td>
<td>66%</td>
</tr>
</tbody>
</table>

sentences. The percentage of Top 1 and Top 3 in one-err have increased more than 20% compared to ErrorError.

Finally is the analysis of the efficiency of the provision of teaching hints. The provision of teaching hints assists students in language learning as argued by Long (1996); Swain (2005). From the provided teaching hints, the students learn and try to correct their ill-formed sentences. Instead, if an answer model of a corresponding ill-formed sentence is simply given to the students, they just copy the correct version without knowing what mistakes they have made. Consider again the results depicted in Table 6.11. The analysis of the provision of teaching hints was performed on three different scopes of ErrorError sentences. The best result yielded was from the analysis on ErrorError sentences which have candidate perturbations only. For 68% of these sentences, my error correction model is able to suggest Top 1 ranked perturbations which can generate effective teaching hints. However, the difference between Top 1 and Top 3 is not big, where the ranking Top 3 is at 75%. What I can conclude here is that, almost 90% of candidate perturbations which lead to effective teaching hints can be generated from, are in the ranking Top 1.

6.4 Summary

In this chapter, I have described the evaluation of my enhancements to the Kaitito system. The goal was specifically to evaluate the performance of my proposed statistical error correction model. Prior to an analysis of evaluation results, I carried out a preliminary checking of each students’ response due to some limitations of the system’s parser. I have performed two types of evaluation: an evaluation of the grammar and an evaluation of the error correction system. Finally, I have discussed the result findings.
As this is the final stage of my research work, the next chapter is the final chapter of my thesis.
Chapter 7

Conclusion

In this chapter, I will first explain my contributions to the area of computer assisted language learning (CALL) systems (§7.1). Then, I describe some possible future directions of the research work as given in §7.2. Lastly in §7.3, I conclude the thesis with a final remark.

7.1 Contributions

The main contribution is the statistical model of error correction I developed (Chapter 4). The main novelty of my statistical error correction model is that it suggests ‘corrections’ for an erroneous sentence, based on what common corrections available for the erroneous sentence. The model was trained on a newly gathered corpus of Malaysian EFL learners’ language data. Then, the model was incorporated into a practical dialogue-based CALL system (Chapter 5) and evaluated by Malaysian EFL learners (Chapter 6). Results from the evaluation show that most of corrections suggested are the most appropriate answer for their corresponding ill-formed sentences in a given context. On top of that, from the derived corrections, the dialogue-based CALL system provides quite good suggestions (teaching hints) about how to correct erroneous sentences.

Secondly, a parser and a grammar were incorporated with the practical dialogue-based CALL system. The system is implemented within the Kaitito system.
is a parser-based CALL system which supports an open-ended conversation, a mixed-initiative and a multi-speaker mode. However, my research focusses on error correction. The system implements the Linguistic Knowledge Builder (LKB) tools as its parser. The LKB tools support the English Resource Grammar (ERG). The ERG is an open source broad coverage grammar developed for parsing and generating systems. Language learners have a smaller grammar knowledge and vocabularies than English native speakers. Therefore, I created a subset of English grammar which is sufficient for the practical system. Besides, I also generate a list of vocabularies that contains Malaysian proper names and places names. The list can be referred by other researchers for their own research.

Third is the learner corpus that I gathered from the empirical study I carried out (Chapter 3). The corpus consists of learners’ personal data, as well as the learners’ sentences both grammatical and ungrammatical, proposed corrections for each ungrammatical sentence, and the related errors. This valuable content can be used by others researchers as a resource for their research work.

Fourth is the error classification scheme I created in the empirical study. The error classification scheme was used as a reference guide during the annotation of errors for the learner corpus. The scheme is statistically valid as I conducted two types of inter-coder reliability test on the scheme: Kappa, $\kappa$ and Alpha, $\alpha$. The results showed a good reliability (for $\alpha$ scale) and a perfect agreement (for $\kappa$ scale) between the two coders. The scheme can become one of error classification scheme resources to be used by researchers.

The fifth contribution is the results of error analysis I conducted on the learner corpus. Results from the analysis provide information to language teachers in Malaysia about the patterns of their students’ errors and how these change over time. Moreover from a longitudinal study I conducted to observe the students’ progress over time, there are some language constructions that the students still need more attention. From my literature review, I highlighted three contributory factors to the students’ errors: an interference of the students’ first language, insufficient exposure to proper English usage, and insufficient time spent for grammar teaching and learning.

Lastly, my other contribution is the simple practical dialogue-based CALL system I
developed (Chapter 5). The system may become a platform for language teachers to be used as a supplementary material to language learning and teaching. Since the teaching pedagogy of English in Malaysia is the communicative approach, this system can be tailored to different types of context. What the teacher needs are a subset of grammar in a given context, a corpus of perturbation sentences and a list of questions to be asked to students. The subset of grammar is need just to cover grammar knowledge for the language learners. The corpus of perturbation sentences consists of a list of pairs of sentences. The first in each pair is an incorrect sentence and the second sentence is the corrected sentence, as judged by the teacher. The corpus must be a set of possible sentences which respond to the list of questions to be asked.

7.2 Future Work

I highlight three possible directions for the future work in this research: the enhancement of the statistical model of error correction, the enhancement of the practical dialogue-based CALL system, and the provision of various types of teaching hints in the practical dialogue-based CALL system.

First is the enhancement of the statistical model. I want to implement one more perturbation action: word transposition. Currently, only three perturbation actions are implemented in my error correction model which are word deletion, word insertion, and word transposition. Another enhancement is to reconfigure the model in such a way that it can provide corrections to an ill-formed sentence which contain more than one error.

Second is the enhancement of the practical dialogue-based CALL system. I want to make the system like a platform system where a teacher can easily create a subset of grammar, a perturbation corpus and a list of questions to be asked in the system based on a content of topic for language learning. The system will automatically generate a $N$-gram perturbation corpus from the perturbation corpus, and load the grammar created. Then the system is then ready to be accessed by students.

The practical dialogue-based system can be configured to be run online or stand alone. If running online, multiple access settings have to be carefully set up to avoid
flaws in flows of dialogue. Another possible enhancement is to extend the system’s parser capability. Currently, the parser is unable to parse a certain sentence which causes the parser to run out of memory. This led the dialogue system become unresponsive. It would be better if the parser is reconfigured to automatically handle such a problem.

Third, is the use of various types of teaching hints in the practical dialogue-based CALL system. This is related to my original research interest. Let me call the teaching hints “corrective feedback” (CF). My original research goal was to further investigate the effectiveness of the provision of corrective feedback in the CALL systems. The provision of CF is proved beneficial to language learning. My statistical model of error correction explicitly gives corrections for an ill-formed sentences. Based on the given corrections, the system can be configured to offer various types of CF i.e. metalinguistic or recast feedback. I would like to conduct an experiment to investigate the effectiveness of the provision of CF and how the CF assist the language learners in correcting their erroneous sentence.

Lastly, I would like to implement Brockett, Dolan, and Gamon’s (2006) machine-translation-inspired approach to error correction using the corpus that I gathered (described in Chapter 3). Brockett, Dolan, and Gamon use synthetic data, not data about real language learner errors. They manually generate synthetic errors with very simple templates a lot like my perturbation corpus: e.g. converting the phrase ‘learn much knowledge’ to ‘learn many knowledge’. My perturbation corpus is generated automatically from a more naturalistic learner corpus that I gathered from real ESL learners. Therefore, the corpus which I gathered would be an ideal one to use with Brockett et al.’s translation based technique.

7.3 Final Remark

First and foremost, the statistical model of error correction I developed can explicitly provide ‘corrections’ for an ill-formed sentence. From the corrections, an effective suggestion about how to correct an error can be provided. Good suggestions are beneficial to language learning.
The dialogue-based CALL system creates more authentic, interesting and motivating learning environments for students to learn a language. The students can use the system any time during their leisure time, not only during school hours but also at home during their leisure time to practise their social conversation. Besides, the language teachers don’t have to supervise their students. The dialogue-based CALL system is also good for shy students who do not want to actively participate or have a low confidence in speaking during the normal language learning in a classroom.

As for the teachers, using the dialogue-based system, they create an alternative way to vary their teaching methods. A variety of teaching methods in language learning will be helpful to increase the students’ interest in learning English and also motivate them to learn more.

The findings from this research will certainly provide fruitful ideas and guidelines to CALL software developers to build more flexible, reliable and interesting CALL systems. Certainly CALL software which incorporates technological and pedagogical approaches are needed. Last but not least, the outcome of the research may also contribute to the Malaysian government and society. As the English language is the second important language used in Malaysia, the dialogue-based CALL system can be used by Malaysians to practise their conversation skills. As the Malaysian government encourages her people to become information technology literate, hopefully the research product will be one resource for Malaysian pupils to acquire and practise their English skills and at the same time be equipped with information and communication technology.
References


Heift, T. and McFetridge, P. (1999). Exploiting the Student Model to Emphasize Language Teaching Pedagogy in Natural Language Processing. In M. B. Olsen (Ed.),


Jurafsky, D. and Martin, J. H. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recog-


Menzel, W. and Schröder, I. (July 2003). Error diagnosis for language learning systems. Lecture notes for a course on Error diagnosis and feedback generation for language tutoring systems at the ELSNET Summer School, Lille, France.


tional ACM conference on Assistive Technologies, Arlington, Virginia, USA, 92–100. Association for Computing Machinery.

Mohideen, H. (1996). Error Analysis - Contributory Facors to Students' Errors, with Special Reference to Errors on Written English. The English Teacher, XXV, –.


Appendix A

Other Classifications for Corrective Feedback

Ferreira (2006) categorises corrective feedback (CF) in Giving-Answer Strategies (GAS) or Prompting-Answer Strategies (PAS) as summarised in Table A.1. The GAS is types of feedback in which the teacher directly gives the target form corresponding to the error in a student’s answer, or shows the location of the student’s error. Repetition and explicit correction feedback are categorised in the GAS group. In contrast, the PAS is types of feedback in which the teacher pushes students to notice a language error in their response and to repair the error themselves. The PAS consists of metalinguistic and elicitation feedback. Ferreira does not mention about translation and paralinguistic signs in her paper.

Besides categorising CF in explicit or implicit form (Ellis et al., 2006), Ellis classifies two more categories: input providing and output providing (Ellis, 2007). The input providing is when the right utterance or “model” is given to students either in explicit

<table>
<thead>
<tr>
<th>Giving-Answer Strategies</th>
<th>Prompting-Answer Strategies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit Correction</td>
<td>Metalinguistic Feedback</td>
</tr>
<tr>
<td>Repetition</td>
<td>Elicitation</td>
</tr>
<tr>
<td>Recast</td>
<td>Clarification Request</td>
</tr>
</tbody>
</table>
Table A.2: Categories of corrective feedback as claimed by Ellis

<table>
<thead>
<tr>
<th>Corrective Feedback</th>
<th>Input Providing</th>
<th>Output Providing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>Explicit Correction</td>
<td>Metalinguistic Feedback</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Elicitation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Paralinguistic Signal</td>
</tr>
<tr>
<td>Implicit</td>
<td>Recast</td>
<td>Repetition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Clarification Request</td>
</tr>
</tbody>
</table>

form or implicit form. On the other hand, output providing feedback give only a hint or information about error done by the students (either implicitly or explicitly) in response to the student’s non-target like utterance. The input providing and output providing strategies are similar respectively to Ferreira’s GAS and PAS as mentioned in the previous paragraph. The summary of explicit/implicit, and input/output providing is depicted in Table A.2.
Appendix B

Teaching and Learning English in Malaysia

The English language is the second language widely used in Malaysia after her national language, *Bahasa Malaysia* or the Malay language. Currently, English is taught as a second language in all Malaysian primary and secondary schools. This section explains briefly approaches as well as various syllabi implemented in teaching and learning English since the independence of Malaysia on 1957 until now. There are two levels of school in Malaysia nowadays: primary and secondary schools. Students spend six years in primary schools and five years in secondary school before continue to higher education. In primary schools, the pupils begin Standard One at seven years old and complete primary education at twelve years old in Standard 6. The student continue secondary education at thirteen years old. The secondary schools have two levels, namely lower level and upper level. The lower level is divided into three forms, namely Form 1, Form 2 and Form 3, whereas Form 4 and form 5 are in upper level secondary school. The students finish secondary education at the age of seventeen years old. Table B.1 below summarizes the level of schools and ages of students attended for each level.

The English language has been introduced in Malaysia (was known as Malaya before independence) by British colony during British colonisation. The Malay language was officially declared as the national language after Independence, but English was still
Table B.1: Summary of Malaysian school levels and ages of students.

<table>
<thead>
<tr>
<th>Schools</th>
<th>Levels</th>
<th>Age of Pupils</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>Standard One</td>
<td>7</td>
</tr>
<tr>
<td>Primary</td>
<td>Standard Two</td>
<td>8</td>
</tr>
<tr>
<td>Primary</td>
<td>Standard Three</td>
<td>9</td>
</tr>
<tr>
<td>Primary</td>
<td>Standard Four</td>
<td>10</td>
</tr>
<tr>
<td>Primary</td>
<td>Standard Five</td>
<td>11</td>
</tr>
<tr>
<td>Primary</td>
<td>Standard Six</td>
<td>12</td>
</tr>
<tr>
<td>Secondary (Lower Level)</td>
<td>Form One</td>
<td>13</td>
</tr>
<tr>
<td>Secondary (Lower Level)</td>
<td>Form Two</td>
<td>14</td>
</tr>
<tr>
<td>Secondary (Lower Level)</td>
<td>Form Three</td>
<td>15</td>
</tr>
<tr>
<td>Secondary (Upper Level)</td>
<td>Form Four</td>
<td>16</td>
</tr>
<tr>
<td>Secondary (Upper Level)</td>
<td>Form Five</td>
<td>17</td>
</tr>
</tbody>
</table>

used as the formal language. Until the year 1970, there were two different school systems operating within Malaysia:

- the national schools which used Bahasa Malaysia as the medium of instruction; and

- the national type English schools which used English exclusively as the medium of instruction.

The national primary schools (non-English medium) used structural syllabus known as The Syllabus for Primary school and Remove Forms (1965). The national secondary schools used The Syllabus for the Secondary Schools (Malay Medium): English (1966) and the national type secondary schools used The Syllabus for the Secondary Schools (English Medium): English (1968).

The implementation of the National Education Policy in the year 1970 resulted in merging the national and national type school systems into one system, with Bahasa Malaysia as the medium of instruction. Therefore, in the new system, English was formally given as the second language status. The Post 1970 Primary English Syllabus
was the first common content syllabus implemented and taught to all standard One primary school pupils. The first batch of the pupils entered Form One classes secondary school in the year 1976. To ensure continuity, the lower secondary syllabus known as *The English Syllabus for Form One - Three (1973)* was developed as an extension of the primary school syllabus. The linguistic content of the syllabus maintained items that have already been covered in the six years of primary education. In 1979, *The English Language Syllabus in Malaysian Schools Form Four - Form Five (1980)* was implemented in the upper forms. While the syllabi for primary and lower secondary school were structural-situational syllabus with an emphasis on oral exercises, the syllabus for upper secondary was a task-oriented situational approach. This upper secondary school syllabus was also called *The Malaysian Communicational Syllabus* (Asiah (1983) as cited in Foo and Rechards (2004)).

The enhancement of the role of Bahasa Malaysia and the corresponding reduction in the role of English led to a decrease in the amount of exposure to English for the students. In addition, Malaysia is a multi races country with races such as Malays, Chinese, Indian and other minority. These people also have their own language such as Malay, Cantonese and Tamil. This is also become one of factors in the reduction of English exposure among Malaysians (Foo and Rechards, 2004).

The existing structural syllabi also had some weaknesses. They focused on discrete learning of grammar. According to Abraham (1987) as cited in Pandian (2002), the structural approach usually provides a list of language structures and words as learning objectives. She further states the language structures are presented orally, normally in a context or situation. Various language drills are employed in teaching new structures. This led to a very restrictive teacher-centred approach. Sentences were learnt in isolation and students who did well in classroom activities found it hard to use the language in a meaningful situation. Besides, the syllabi were designed more for students who were constantly exposed to English. A little consideration was paid to students from non-English speaking backgrounds (Rajaretnam and Nalliah (1999) cited in Foo and Rechards (2004)). As a result, students whose background was not English speaking or who were from the rural areas left the education system with very low proficiency in English.
Due to these discrepancies, the syllabi were reviewed and analysed by the English Language Renewal Committee under the guidance of the Curriculum Development Centre. As a result, the Curriculum Development Centre designed two new syllabi which are implemented in primary and secondary schools until now. The implementation of the New Primary Schools Curriculum or *Kurikulum Baru Sekolah Rendah (KBSR)* in 1983 and the Integrated Secondary Schools Curriculum or *Kurikulum Bersepadu Sekolah Menengah (KBSM)* in 1989 was a step forward in the implementation of an education system with a common goal, direction and approach. The KBSR aims to equip learners with basic skills and knowledge of the English language so as to enable them to communicate, both orally and in writing, in and out of school (Kementerian Pendidikan Malaysia, 2001). After six years of primary education through the KBSR, students continued their secondary education under a implementation of KBSM syllabus. The aim of the KBSM curriculum is to equip students with communicative ability and competency to perform language functions, using correct language forms and structures (Kementerian Pendidikan Malaysia, 2000). The KBSR and KBSM syllabi are designed to carry out communicative activities in the classroom. Examples of such activities are games, drama, simulation, and projects which make use of English in realistic and contextualized situations. These activities involve doing things with the language, such as making choices, evaluating and bridging the information gap. The contents of the KBSR and KBSM syllabi are arranged according to topics. The topics for KBSR are World of Family and Friends, World of Stories and World of Knowledge. As for the KBSM, the topics to be taught for each year are organised carefully according to five main themes, namely People, Social Issues, Science and Technology, Environment, and Health. Please refer to Kementerian Pendidikan Malaysia (2003a,b) for further detailed syllabus for each year of secondary and primary schools.

The teaching and learning English language in Malaysia has undergone several phases. It began with the implementation of structural-situational approach since the British colonisation until 1983. Due to some weaknesses of the methodology, a different approach called a communicative approach is introduced. While the structural-situational methodology focused on learning of rules, the communicative approach instead, emphasises on the use of English in a meaningful situation based on local
cultural and environment. In conjunction to the changes of methodology, the syllabi also have been revised. The KBSR and KBSM is the current curriculum for teaching and learning English used in all schools in Malaysia.
Appendix C

The Questions featuring in the Student Questionnaire

C.1 Present Tense Questions
**Please answers all questions in complete sentences.**

**Present tense grammar**

<table>
<thead>
<tr>
<th></th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What is your name?</td>
<td>My name (be)...</td>
</tr>
<tr>
<td>2</td>
<td>Which country are you from?</td>
<td>I (be)...</td>
</tr>
<tr>
<td>3</td>
<td>Tell me more about your country.</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Which city are you from?</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Tell me about your city.</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>How old are you?</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>What form are you now?</td>
<td>.......... in Form 1 / 2 / 3.</td>
</tr>
<tr>
<td>8</td>
<td>What do you like doing?</td>
<td>I like...</td>
</tr>
<tr>
<td>9</td>
<td>How many brothers and sisters do you have?</td>
<td>I have...</td>
</tr>
<tr>
<td>10</td>
<td>What is your best friend’s name?</td>
<td></td>
</tr>
<tr>
<td>Question</td>
<td>Answer</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>How many brothers and sisters does your best friend have?</td>
<td>She/He ....</td>
<td></td>
</tr>
<tr>
<td>How old is your best friend?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What does your best friend like doing?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What do your parents like doing?</td>
<td>They like ...</td>
<td></td>
</tr>
<tr>
<td>Describe your parents.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>What do you and your friends do together?</td>
<td>We ...</td>
<td></td>
</tr>
<tr>
<td>What is your father's job?</td>
<td>He (be) ...</td>
<td></td>
</tr>
<tr>
<td>Describe what your father does in his job.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### C.2 Past Tense Questions

**Please answers all questions in complete sentences.**

**Past tense grammar**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Where were you born?</td>
</tr>
<tr>
<td></td>
<td>I...</td>
</tr>
<tr>
<td>2</td>
<td>What was the best thing you did today?</td>
</tr>
<tr>
<td>3</td>
<td>Why did you like it?</td>
</tr>
<tr>
<td>4</td>
<td>What was the worst thing you did today?</td>
</tr>
<tr>
<td>5</td>
<td>Why didn't you like it?</td>
</tr>
<tr>
<td>6</td>
<td>What did you do last weekend?</td>
</tr>
<tr>
<td></td>
<td>I...</td>
</tr>
<tr>
<td>7</td>
<td>What did your parents do last weekend?</td>
</tr>
<tr>
<td></td>
<td>They...</td>
</tr>
<tr>
<td>8</td>
<td>What did your best friend do last weekend?</td>
</tr>
<tr>
<td>9</td>
<td>Where did you go on your last school holiday?</td>
</tr>
<tr>
<td>10</td>
<td>Who went with you on your holiday?</td>
</tr>
</tbody>
</table>

Please turn over...
| 11 | What did you like most about your school holiday? |
| 12 | Where did your best friend go on his/her holiday? |
|     | She / He ... |
| 13 | Where did you celebrate Hari Raya Aidil Fitri/ Chinese New Year / Deepavali / Christmas day? (choose one) |
|     | I ... |
| 14 | What did you and your family do on Hari Raya Aidil Fitri/ Chinese New Year / Deepavali / Christmas day? (choose one) |
|     | We ... |
C.3 Future Tense Questions

Please answers all questions in complete sentences.

**Future tense grammar**

<table>
<thead>
<tr>
<th></th>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What will you do when you leave this class?</td>
<td>I...</td>
</tr>
<tr>
<td>2</td>
<td>What are you going to do this evening?</td>
<td>I...</td>
</tr>
<tr>
<td>3</td>
<td>What will your parents do this evening?</td>
<td>They...</td>
</tr>
<tr>
<td>4</td>
<td>What is your best friend going to do this evening?</td>
<td>She/He...</td>
</tr>
<tr>
<td>5</td>
<td>What do you want to do next weekend?</td>
<td>I...</td>
</tr>
<tr>
<td>6</td>
<td>What will your best friend do next weekend?</td>
<td>She/He...</td>
</tr>
<tr>
<td>7</td>
<td>What will your parents do next weekend?</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Where will you celebrate the next Hari Raya Aidil Fitri / Chinese New Year / Deepavali / Christmas? (choose one)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>What will you and your class study during English classes next week?</td>
<td>We...</td>
</tr>
</tbody>
</table>

Please turn over...
<table>
<thead>
<tr>
<th></th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Where are you and your friends going to play during recess?</td>
</tr>
<tr>
<td></td>
<td><em>We ...</em></td>
</tr>
<tr>
<td>11</td>
<td>What do you want to be when you grow up? <em>(What is your ambition?)</em></td>
</tr>
<tr>
<td></td>
<td><em>I ...</em></td>
</tr>
<tr>
<td>12</td>
<td>What do you have to do to achieve your ambition?</td>
</tr>
<tr>
<td>13</td>
<td>What is your best friend's ambition?</td>
</tr>
<tr>
<td></td>
<td><em>She/He ...</em></td>
</tr>
</tbody>
</table>
Appendix D

Tags of Error Categories
<table>
<thead>
<tr>
<th>Error Category</th>
<th>Error Tag</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alteration</td>
<td>insert(X)</td>
<td>Delete X to fix a sentence.</td>
</tr>
<tr>
<td></td>
<td>delete(X)</td>
<td>Insert X to fix a sentence.</td>
</tr>
<tr>
<td></td>
<td>subst-with(X, Y)</td>
<td>Substitute X with Y to fix a sentence.</td>
</tr>
<tr>
<td></td>
<td>transpose(X, Y)</td>
<td>Swap adjacent X and Y to fix a sentence.</td>
</tr>
<tr>
<td>Agreements</td>
<td>va(X)</td>
<td>X refers to a verb which does not agree with a subject/noun in a sentence.</td>
</tr>
<tr>
<td></td>
<td>det-n-ag</td>
<td>No agreement between a determiner and a noun.</td>
</tr>
<tr>
<td></td>
<td>noun-num-err</td>
<td>A noun in a sentence which is not represented as a generic noun word.</td>
</tr>
<tr>
<td></td>
<td>tense-err(A, B)</td>
<td>Use B tense form instead of A tense form.</td>
</tr>
<tr>
<td>Options</td>
<td>no-resp</td>
<td>No response.</td>
</tr>
<tr>
<td></td>
<td>part-resp</td>
<td>Partial response.</td>
</tr>
<tr>
<td></td>
<td>incorp-resp</td>
<td>Incomplete responses to the given questions.</td>
</tr>
<tr>
<td></td>
<td>All non-English words including mis-spelled ones.</td>
<td></td>
</tr>
<tr>
<td>Spell-err</td>
<td>spell-err</td>
<td>Mis-spelled English words.</td>
</tr>
<tr>
<td>Voc-err</td>
<td>vocab-err</td>
<td>All non-English words including mis-spelled ones.</td>
</tr>
<tr>
<td>Dialogues</td>
<td>unrel-ans</td>
<td>Irrelevant answers to its given questions.</td>
</tr>
<tr>
<td></td>
<td>incomp-ans</td>
<td>Incomplete responses.</td>
</tr>
<tr>
<td></td>
<td>part-ans</td>
<td>Partial responses.</td>
</tr>
<tr>
<td></td>
<td>no-resp</td>
<td>No response.</td>
</tr>
<tr>
<td>Unclassification</td>
<td>unclsfid</td>
<td>A response has more than four error codes or does not fall into one of error categories.</td>
</tr>
</tbody>
</table>
Appendix E

A Complete Set of Sentences used to develop the reduced-ERG
Figure E.1: A complete set of sentences used to create the reduced ERG grammar, continued next page
("He teaches people." 1) ("My father sells food to the students." 5) ("He is teaching in London." 1) ("I went to London." 2) ("I went to London last weekend." 7) ("I'll celebrate it at my house." 1) ("I will celebrate Christmas in London." 4) ("I will celebrate the next Christmas in London." 1) ("I will celebrate merry Christmas with my friend." 1) ("I will celebrate the next Christmas in my house." 1) ("I celebrated Christmas day at home." 3) ("I hope I celebrate in our house." 1) ("She may play video games." 1) ("She will do revision." 1) ("She will attend her tuition classes." 1) ("She will study with me." 1) ("We will study about articles." 1) ("We are going to play football." 1) ("I want to visit my family." 1) ("I never think about it." 1) ("She will finish her homework." 1) ("He will study at home." 1) ("I will be doing my homework." 1) ("I will ask my teacher." 2) ("We collected money." 1) ("We visited our relatives." 1) ("We celebrated together." 1) ("I like to watch television." 1) ("I went with my parents." 1) ("I didn't do anything." 1) ("She also watched television." 1) ("He just sat at home." 1) ("She gave me a present." 1) ("He was absent." 1) ("I slept." 1) ("I was playing the gameboy." 1) ("I was late to school." 1) ("I was very angry." 1) ("It is a wrong thing." 1) ("I made a noise and a girl scolded me." 2) ("I did nothing." 1) ("Nothing is considered worst." 1) ("It was a good action." 1) ("It is a good activity." 1) ("I like to play computer games." 1) ("I like drawing pictures." 5) ("I like to read books." 1) ("I like to repair my brother's bicycle." 1) ("I like reading a book." 2) ("I like to do revision in the afternoon." 1) ("I like to listen to the radio." 1) ("I like to listen to pop music." 1) ("I like cycling the bicycle at the park." 1) ("I like to read story books." 1) ("I like to do something adventurous." 1) ("I really like reading the comics." 1) ("I love listening to music." 8) ("I can release my tension." 1) ("I sang with my best friend." 1) ("I like to cook." 1) ("We like reading together." 8) ("I have lied to my friend." 2) ("They sent me to school." 1) ("They went out." 2) ("She took me to London." 1) ("He baked me a chocolate cake." 1) ("She visited her aunt last weekend." 3) ("I went to the shopping mall." 1) ("I did not do my homework." 2) ("I played tennis with my best friend." 1) ("I played with my pet." 2) ("He enjoyed playing tennis with me." 9) ("I went to my grandfather's house." 1) ("I went there with my cousins." 1) ("I went there alone." 1) ("I went to Thailand on last school holiday." 4) ("My parents came with me." 2) ("I went there with my relatives." 1) ("They went to a shopping complex." 3) ("They went to the wedding reception." 3) ("She went for a trip to China." 1) ("She went to the camping." 2) ("I celebrated the Christmas in my house." 3) ("My father is a kind and honest man." 17) ("My mother is a polite and beautiful woman." 2) ("My parents are caring and strict." 2) ("They are a romantic couple." 1) ("My parents are nice and good parents." 5) ("They like to tidy up my house." 2) ("They like playing bowling." 3) ("They like shopping." 1) ("They like to plant flowers." 1) ("They like to spend time with their daughter." 3) ("They like to travel." 1) ("Now I study in school." 1) ("Now, I'm in school." 1) ("I'm in school." 1) ("I did my school work." 1) ("I will eat food when I leave this class." 7) ("I like it because I can have knowledge." 1) ("I like it because it is my hobby." 1) ("I like it because that was my passion." 1) ("I liked it because I learned a new chapter." 1) ("I didn't bring my English textbook." 1) ("I didn't like the cat." 1) ("I didn't like ill mannered people." 1) ("I went camping in the school." 4) ("My best friend went to a birthday party." 1) ("We must do a lot of study." 1) ("I could play with my pet." 1) ("I can sleep everyday." 2) ("I might sleep." 1) ("We enjoyed playing fire crackers." 10) ("I will go straight home." 1) ("I will wash my clothes." 1) ("I will sleep on my bed." 1) ("My father is tall and thin." 6) ("They were very nice and friendly." 1) ("They cooked my favourite dishes." 3) ("He passed away several years ago." 2) ("We will play basketball during recess." 2) ("I want to be a doctor when I grow up." 29) ("I didn't like it because it hurt my friend's heart." 1) ("I'm free to enjoy myself." 1) ("We will study grammar during English classes next week." 10) ("He goes to London with me." 5) ("He teaches pupils about History." 2) ("My father is the head of a project." 4) ("We play basketball every Saturday." 1) ("They like to spend their time with their kids." 3) ("My best friend likes to read books and novels." 2) ("They like to talk to each other." 6) ("My city has many food stalls." 3) ("Because I like to cook." 1) ("Because it is my favourite subject." 7) ("Because it made me look bad." 7) ("My parents." 2) ("My family and I." 1) ("Everything is wonderful." 1) ("We visited our relatives and treated guests who came to our house." 11) ("They will talk about ourselves." 7))
Appendix F

A Listing of the Additional Malaysia-specific Proper Names added to Kaitito’s lexicon
Figure F.1: Malaysian people's names
Figure F.2: Malaysian places names
Appendix G

Students’ Transcript Samples

G.1 Sample 1
-----Start Dialogue-----
ID: 110331151202-NQEHB2DHCKMG
1) What is your name?
Hmm, your response may look like this: My name is _____.
1) What is your name?
You> reading book
That's not quite right! May be you have to
1: replace the word 'book' with 'books' in your answer.
2: use the word 'the' in your sentence.
3: use the word 'a' in your sentence. Please try again.
1) What is your name?
2) Which city are you from?
You> my best friend name is jung min
That's right!
3) How old are you?
You> I born at Hospital Melaka
That's not quite right! Perhaps you could
1: use the word 'was' in your response.
2: use the word 'am' in your response. Please try again.
3) How old are you?
3) How old are you?
You> i was born at hospitl melaka
That's it!
4) Where were you born?
You> I born in Malacca.
That's not quite correct! May be you could
1: use the word 'was' in your response.
2: use the word 'am' in your answer. Please try again.
4) Where were you born?
Well, your answer may look like this: I was born in _____. Please try again.
4) Where were you born?
You> My bestfriend name is Nur Fadilla
Well done!
---Start Dialogue---
ID: 110331175641-HNS3CD82GH6K
1) What is your name?
You> my name is muhd afif jazimin bin mat ali
Nice one!
2) Which city are you from?
3) How old are you?
You> i born at melaka
That's not quite correct! Perhaps you could
1: use the word 'was' in your response.
2: use the word 'am' in your response. Please try again.
3) How old are you?
You> my name is Nurul Anis Nashirah
Yes, that's right!
4) Where were you born?
You> i'm from perak
That's it!
5) How many brothers and sisters do you have?
You> I have two brothers and two sisters
Good!
6) What do you like doing?
You> i like play rugby
Good!
7) What is your best friend's name?
You> i'm born at hospital kampar,perak
That's right!
8) How many brothers and sisters does your best friend have?
You may respond as: My best friend/She/He has __ brothers and ___ sisters.
8) How many brothers and sisters does your best friend have?
You> 6
Great answer!
9) How old is your best friend?
9) How old is your best friend?
You> I like playing badminton
Nice one!
10) What does your best friend like doing?
You may respond as: My best friend/She/He likes ________.
10) What does your best friend like doing?
You> 14 year old
That's not quite right! May be you have to
1: replace the word 'year' with 'years' in your answer.
2: use the word 'years' in your sentence.
3: use the word 'age' in your response. Please try again.
10) What does your best friend like doing?
You> 14 years old
Well done!
11) What do your parents like doing?
You> my best friend age 14 year old
That's not quite right! Perhaps you need to
use the word 'age.' in your response. Please try again.
11) What do your parents like doing?
G.3 Sample 3

---Start Dialogue---
ID: 110331164421-SLTRK6EM2TS0
You may respond as:
1) What is your name?
You> cute
Nice one!
2) Which city are you from?
3) How old are you?
You> My parent like gardening
That's not quite right! Perhaps you have to
1: replace the word 'parent' with 'parents' in your answer.
2: use the word 'i' in your answer.
3: replace the word 'like' with 'liked' in your sentence. Please try again.
4) Where were you born?
You> boling
Nice one!
5) How many brothers and sisters do you have?
You> facebook
Great answer!
6) What do you like doing?
You> play
Awesome!
7) What is your best friend's name?
You> na inang
Good!
8) How many brothers and sisters does your best friend have?
You> chicharito
Awesome!
9) How old is your best friend?
You> pikat
Good!
10) What does your best friend like doing?
You> ok
Well done!
11) What do your parents like doing?
You> gdghytr
That's right!
12) What do you and your friends do together?
You> fsrgth
Wonderful!
13) What is your father's job?
Well, your answer may look like this: My father's/His job is ______.
14) What did you do last weekend?
You> hjdfyj
Great answer!
15) What did your best friend do last weekend?
You> my best friend solder
That's not quite correct! Perhaps you have to replace the word 'friend' with 'friend's' in your sentence. Please try again.
16) What did your best friend do last weekend?