Grammatical Error Identification for Learners of Chinese as a Foreign Language

Yang Xiang

Uppsala University
Department of Linguistics and Philology
Master’s Programme in Language Technology
Master’s Thesis in Language Technology
September 27, 2018

Supervisors:
Joakim Nivre, Uppsala University
Abstract

This thesis aims to build a system to tackle the task of diagnosing the grammatical errors in sentences written by learners of Chinese as a foreign language with the help of the CRF model (Conditional Random Field). The goal of this task is threefold: 1) identify if the sentence is correct or not, 2) identify the specific error types in the sentence, 3) find out the location of the identified errors.

In this thesis, the task of Chinese grammatical error diagnosis is approached as a sequence tagging problem. The data and evaluation tool come from the previous shared tasks on Chinese Grammatical Error Diagnosis in 2016 and 2017. First, we use characters and POS tags as features to train the model and build the baseline system. We then notice that there are overlapping errors in the data. To solve this problem, we adopt three approaches: filtering out the problematic data, assigning encoding to characters with more than one label and building separate classifiers for each error type. We continue to increase the amount of training data and include syntactic features. The results show that both filtering out the problematic data and including syntactic features have a positive impact on the results. In addition, difference between domains of training data and test data can hurt performance to a large extent.
Contents

Preface 4

1 Introduction 5
  1.1 Purpose 5
  1.2 Outline 5

2 Background 7
  2.1 Automated Grammatical Error Detection For English 7
    2.1.1 Rule-based Approaches 7
    2.1.2 Data-driven Approaches 9
    2.1.3 Recent Grammatical Error Correction Shared Tasks 11
  2.2 Chinese Grammatical Error Detection 11

3 Task Data and Tools 14
  3.1 Task and Data 14
  3.2 Conditional Random Field 17
  3.3 Stanford CoreNLP 18

4 Experiments 20
  4.1 Baseline System 20
  4.2 Modified Systems 22
    4.2.1 Dealing with Overlapping Errors 22
    4.2.2 Increasing the Amount of Training Data 23
    4.2.3 Adding Syntactic Features 24
  4.3 Final Test Results 25
  4.4 Discussion 26

5 Conclusion 28
  5.1 Conclusion 28
  5.2 Future Work 29

Bibliography 29
Preface

First of all, I would like to thank my supervisor Joakim Nivre for all the kind help as well as the brilliant advice and ideas he has provided.

Moreover, I am grateful to all the teachers and fellow students of the Department of Linguistics and Philology for their dedication and support throughout the two years.

Last but not least, I want to thank my dear family and friends for always being there for me no matter what.
1 Introduction

More and more people are taking an interest in learning Chinese nowadays as a result of globalisation. What makes Chinese relatively more difficult to learn than other languages such as English is that it has a loose structure and tends to use more short sentences instead of complex clauses. The logic between these short sentences is weak and some information remains somewhat implicit or unsaid. In addition, Chinese does not differentiate between singular form and plural form. Also, no change of verb tense is displayed (Xie et al., 2017). Therefore, Chinese learners are in need of grammatical correction tools to help them master the language. However, since the CFL (Chinese as a foreign language) data is quite limited, it was not until 2014 that a shared task on grammatical error diagnosis for learners of Chinese as a foreign language was organized by the ICCE (International Conference on Computers in Education) 2014 workshop on NLP-TEA (Yu et al., 2014). Similar shared tasks were proposed for the following three years.

1.1 Purpose

The aim of this thesis is in line with that of the latest shared task in 2017 (Gaoqi et al., 2017), which is to tackle the task of Chinese grammatical error diagnosis in learner language on three levels:

1. Identify if the sentence is correct or not.
2. Identify the specific error types in the sentence.
3. Identify the exact location of the identified errors.

We approach this task using Conditional Random Fields (CRF) (Lafferty et al., 2001) and attempt to answer the following questions in this thesis:

- How should we deal with the overlapping errors in the training data?
- Does increasing the training set size benefit the result?
- Does adding dependency labels improve the result?

1.2 Outline

The rest of the thesis is structured as follows:

- Chapter 2 focuses on reviewing the background and related work on the task of Chinese grammatical error identification.
• Chapter 3 introduces the datasets and tools used in this paper. We explain how the training data is annotated and provide a detailed introduction to the CRF model.

• Chapter 4 elaborates on the baseline system and various experiments which are conducted to attempt to improve the baseline system and discusses the experimental results. We select the system that produces the best results based on the development set and apply it to the test set to obtain the final experimental results.

• Chapter 5 summarizes the main findings of the paper and makes suggestions for future work.
2 Background

As more and more people decide to pick up a second language, the need for automated grammatical error detection has been rising rapidly. Among all the languages, English is the most popular choice for a second language. Guo and Beckett (2007) point out that English is spoken by over a billion people as their second language. According to Leacock et al. (2014), there are substantially more NLP tools dedicated to dealing with English. Furthermore, a huge amount of data produced by native English speakers is available to be used for training. Consequently, the advances in grammatical error detection tend to be more significant for English than for other languages. However, generally speaking, automated error detection for language learners still remains an underrepresented research topic.

2.1 Automated Grammatical Error Detection For English

The earliest tools commonly used for grammar checking such as Unix Writer’s Workbench (Macdonald et al., 1982) were built on string matching instead of linguistic analysis. As time went by, more and more tools began to integrate linguistic analysis into their grammar checking systems. For instance, both IBM’s Epistle (Heidorn et al., 1982) and Critique system (Richardson and Braden-Harder, 1988) took advantage of complicated grammars and parsers to implement linguistic analysis. Nevertheless, data-driven methods have gradually taken the place of rule-based approaches in grammatical analysis since 1990s. Parsers were developed by incorporating grammatical rules that are written by linguists until large annotated treebanks such as the Penn Treebank (Marcus et al., 1993) and the first statistical parsers (Charniak, 1996; Magerman, 1995) which are trained using treebanks came along.

2.1.1 Rule-based Approaches

Grammatical error detection and correction tools that are based on rule-based approaches require the grammars used to be able to tolerate errors. It is essential that sentences containing common grammatical errors such as subject-verb disagreement can be parsed so that the error correction task can be successfully performed. Apart from this, the parser should also be capable of providing sufficient information in the final parse to show that certain grammatical rules have indeed been violated. Large-scale grammars that meet the above mentioned requirements include Augmented Phrase Structure Grammar (Heidorn et al., 1982), Constraint Grammar (Karlsson et al., 1995) and also grammars which are based on Lexical Functional Grammar (Butt et al., 2002; Frank et al., 1998; Reuer, 2003).
Leacock et al. (2014) divides methods that enable grammars to be error-tolerant into five categories:

1. Generate a number of parse trees and rank them according to the degree with which they conform to the grammatical rules. The parse that breaks the least number of rules is considered to be the best result (Dini and Malnati, 1993).

2. Restrict the number of generated parse trees by attributing different weights to specific violations of grammatical rules or even establishing that certain grammatical rules are not allowed to be violated (Heinecke et al., 1998; Menzel, 1990).

3. Introduce mal-rules or incorrect rules in order to enable the parser to deal with input containing certain errors. The concept of mal-rules was first introduced by Sleeman (1984) and was meant for errors student made in their studies on math. Schneider and McCoy (1998) adopt the mal-rule method with language learners as their target. Bender et al. (2004) divide mal-rules into syntactic mal-rules and two types of lexical mal-rules. They also state that mal-rules can benefit language learners on the condition that the frequency of targeted error types is high, although mal-rules do tend to hurt recall since they are only designed to cover specific error types.

4. Relax certain grammar constraints so that the parser still works even if there are conflicting elements. Works on grammar relaxation include Kwasny and Sondheimer (1981), Douglas and Dale (1992), Hagen (1995) and so on.

5. Enlarge a grammar with rules that attempt to fit together fragments of a parse tree when no complete parse can be produced based on the input. Jensen et al. (1992) put forward a fitting process which takes action when a bottom-up parser cannot generate a root node. Mellish (1989) proposes another way to deal with an unsuccessful bottom-up parser, which is to run a top-down parser afterwards.

However, none of the above mentioned approaches has the capability to analyze random ungrammatical input. Another problem is the possibly significant large amount of errors in learner writing which pose great difficulty to parsers and taggers. For example, systems like the Scripsi system (Catt and Hirst, 1990) and the Intelligent Language Tutor (Schwind, 1990) were developed based on limited grammars and were only able to target a specific range of errors. Generally speaking, error detection or correction tools that specifically target learner writing were quite limited at that time.

An obvious alternative of developing tools for learner error detection of correction is to make use of the existing tools designed for native speakers such as Microsoft Word. Nevertheless, this option is bound to bring about problems. To begin with, since the tools are tailored for native speakers instead of language learners, the errors they target cannot fully cover the wide range of learner errors. Furthermore, the distinctly different nature of errors made by native speakers and learner errors can also cause the tools to underperform. Bolt (1992)
evaluated eight proofreading tools on several learner errors and confirmed that all the tools did not perform well on the learner errors.

2.1.2 Data-driven Approaches

Compared to rule-based methods, one major advantage that data-driven approaches or statistical approaches possess is that they do not have to deal with the error intolerance problem rule-based approaches have. Statistical approaches are able to tackle words that are connected in any sequence since choice of words and different sequences of words are assigned different possibilities according to the frequency at which they have been observed during training.

The idea of tackling grammatical error detection task with statistical classifiers originated from the research on word sense disambiguation (Leacock et al., 2014). Gale et al. (1992) propose to use Bayesian classifiers to solve the task of word sense disambiguation. The main difficulty that researchers run into is that the word sense disambiguation task relies heavily upon manually annotated corpora which were time-consuming and expensive to construct. To overcome this difficulty, Yarowsky (1994) suggests a solution to the task of restoring accent in Spanish and French which is quite similar to the task of word sense disambiguation without depending on manually annotated corpora. First, he trains a decision list classifier based on each word in the sentence that contains the accented words. He then goes on to get rid of the accents of the accented words and uses the decision list classifier to predict whether certain words should be accented in one way or another. Inspired by Yarowsky’s idea, Golding (1995) and Golding and Roth (1996) use decision lists and Bayesian classifiers to identify common spelling errors that come into existence as a result of confusing words that have the same or similar sounds. They build models for each word in a confusion set based on the context in which they are correctly used. A word will be identified as an error if its context is proved to be more similar to the context of the other word in the confusion set. Nonetheless, this approach can not be applied to grammatical error detection in a general sense because spelling errors that result from confusing words that have the same or similar sounds can only make up a small part of grammatical errors. When it comes to grammatical errors such as article errors or preposition errors, it would be too costly to construct models for the usage of each article or preposition in English. According to Leacock et al. (2014), most of the research work of automatic grammatical error detection focus on predicting article error and preposition errors, and approaches that are adopted to predict article error and preposition errors can be categorized based on three criteria: features extracted, training data used and models constructed. Leacock et al. (2014) describes these criteria in details as follows:

Features extracted from the training data fall into the following categories:

- Neighboring tokens within a certain window size are extracted as features to train the classifier.
- Syntactic context is also commonly used as features. Both POS tags and parse information which comes from either constituency parsing or dependency parsing belong to syntactic context.
Semantic information can be useful features as well. For instance, if a noun turns out to be a kind of organization, it is highly likely to co-occur with the definite article.

Source information such as the learner’s native language plays a crucial role in grammatical error identification since taking such background information into account can better target specific errors.

When it comes to training data, there are generally three kinds of training data that are used by most statistical approaches.

1. Training on entirely grammatical text. Since learner corpora are not always easily accessible, systems are trained on native corpora instead. Widely used native corpora include Gigaword Corpus\(^1\), the Google n-gram corpus\(^2\) and so on. One major problem of using native corpora as training data is that even though the size of these corpora is gigantic, the style of them is almost always completely different from that of the learner writing.

2. Training on grammatical text and artificial errors. Artificial errors are generated to compensate for the lack of learner corpora. Rozovskaya and Roth (2010) explore four examples of producing artificial errors and prove that training on text containing artificial errors actually performs better than training on entirely correct text.

3. Training on grammatical text and manually annotated learner corpora. As time goes by, annotated corpora become more and more readily accessible, making it possible for models to be trained on both native and learner corpora. Han et al. (2010) discover that a model for preposition error identification trained on both native and learner usage greatly outperforms model trained solely on correct text.

As for different models or methods that are used for error identification and correction, they generally fall into two categories:

1. Classification is a commonly used method for error detection and correction. Information extracted from the context will be used as features to be learnt by a classifier during training.

2. Language modeling is another well-studied approach. According to Jurafsky and Martin (2009), language models have proved to be very successful in various aspects of natural language processing. They are trained on large-sized corpora where n-grams are counted. The probability of words that come in any sequence can then be calculated based on these n-gram counts. Smoothing techniques are used to tackle words of unseen sequences by assigning a small probability to them to prevent zero probabilities.

\(^1\)https://catalog.ldc.upenn.edu/LDC2003T05
\(^2\)https://catalog.ldc.upenn.edu/LDC2006T13
2.1.3 Recent Grammatical Error Correction Shared Tasks

Past research on grammatical error detection and correction has suffered from lack of learner corpora and shared evaluation metrics. Fortunately, efforts have been made by the community to solve this problem recently. The year 2011 witnessed the first shared task named Helping Our Own (HOO). The idea of this shared task was first put forward by Dale and Kilgarriff (2010). The 2011 shared task (Dale and Kilgarriff, 2011) centers on facilitating the development of automated writing assistance tools for non-native computational linguists within the natural language processing community. The methods adopted by the participants of the shared task consist mainly of heuristic approaches and data-driven approaches. Based on the results obtained by the participants, it can be seen that data-driven approaches are able to yield satisfying results on preposition and article errors. In addition, using the ACL anthology as the corpus instead of corpora from other sources to generate n-gram counts for correct usage proves to play a significant role in creating a successful system. One year later, another HOO shared task was organized. The 2012 shared task (Dale et al., 2012) aims solely at preposition and article errors. A major difference between the 2011 and 2012 shared tasks is that almost all the participants employed data-driven approaches in the 2012 shared task.

In 2013, yet another shared task was dedicated to solve the problem of grammatical error correction. It was conducted along with the Seventeenth Conference on Computational Natural Language Learning (CoNLL). Compared to previous HOO shared tasks, the 2013 CoNLL shared task (Ng et al., 2013) is not limited to only dealing with preposition and article errors but also attempts to tackle other errors such as subject-verb agreement errors, noun number errors and verb form errors. Approaches used include language modeling, classification, heuristic rules and machine translation. The 2014 CoNLL shared task (Ng et al., 2014) takes one step further to devote itself to dealing with all error types instead of only five error types as in the 2013 CoNLL shared task. Since all error types are targeted in this shared task, a majority of the participants adopt hybrid systems that combine different approaches.

To sum up, the above mentioned shared tasks have contributed significantly to the advancement of grammatical error detection and correction by providing corpora and evaluation metrics and attracting attention in the field of natural language processing.

2.2 Chinese Grammatical Error Detection

As a result of globalisation, more and more people become interested in learning Chinese. What makes Chinese relatively more difficult to learn than other languages such as English is that it has a loose structure and tends to use more short sentences instead of complex clauses. Furthermore, Chinese does not differentiate between singular form and plural form. Also, no change of verb tense is displayed (Xie et al., 2017). Consequently, learners of Chinese as a foreign language are urgently in need of grammatical correction tools to help them. Nevertheless, compared to the large amount of research concerning grammatical error detection and correction for English learners, much fewer
studies focus on grammatical error diagnosis for learners of Chinese as a foreign language. These studies put forward a number of approaches to deal with Chinese grammatical error identification. Statistical approaches is one of the commonly adopted methods. For example, C.-H. Wu et al. (2010) propose to employ both a relative position language model and a parse template language model to handle the grammatical detection problem, especially the word order errors. R.-Y. Chang et al. (2012) suggest a penalized probabilistic first-order inductive learning algorithm and decomposition-based testing mechanism to diagnose errors. A linguistic rule-based method is employed by Lee et al. (2013) to identify learner errors. Lee et al. (2014) put forward a hybrid method which integrates a rule-based approach and an n-gram statistical method in order to handle grammatical error diagnosis.

The reason that there is not much research on this topic is largely due to the lack of available CFL (Chinese as a foreign language) data. To cope with this challenge, a shared task on grammatical error diagnosis for learners of Chinese as a foreign language was organized by the ICCE (International Conference on Computers in Education) 2014 workshop on NLP-TEA (Natural Language Processing Techniques for Educational Applications) (Yu et al., 2014). Similar shared tasks were proposed for the following three years consecutively (Gaoqi et al., 2017; Lee, Gaoqi, et al., 2016; Lee et al., 2015). All these shared tasks have the same goal of diagnosing grammatical errors but differ in requirements. There are basically four types of errors to be detected: R (redundant), M (missing), W (word order) and S (selection). The 2014 shared task only requires the participants to identify the error type in the sentences. The given sentences are either correct or contain only one certain error. As for the 2015 shared task, the given sentences also contain only one error, but the participants are required to identify the type of the error and also the position at which they occur in the given sentences. For the 2016 and 2017 shared tasks, the given sentences may contain more than one error, which makes it more difficult to achieve the same goal as the previous shared task.

Generally speaking, approaches that were used in the shared tasks varied greatly from rule-based systems to neural network approaches. T.-H. Chang et al. (2014) propose a rule-based system containing both hand-crafted and machine generated rules to diagnose the errors in learner sentences. Statistical methods are also commonly used. Zampieri and Tan (2014) put forward a frequency-based approach which compares the n-gram frequency in the correct usage corpus to that of the learner corpus in order to detect errors. Statistical machine translation methods are also employed to achieve the goal of detecting grammatical errors (Zhao et al., 2014, 2015). X. Wu et al. (2015) attempt to build a hybrid model by combining rule-based system and n-gram based statistical method. Supervised machine learning methods are also studies extensively. For example, Xiang et al. (2015) experiment with ensemble classifiers as well as several single classifiers such as decision tree, support vector machines and so on. According to the results, ensemble classifiers perform better than single classifiers when it comes to dealing with grammatical error identification. Another two commonly studied approaches are the Conditional Random Fields (CRF) model and Long Short-Term Memory (LSTM) networks. CRF is considered to be the more traditional approach and requires careful feature engineering. Participants experiment with features such
as POS tags, collocation features and word order sensitive embedding features (Chou et al., 2016; PO-LIN et al., 2016; Shih-Hung et al., 2015). Compared to CRF, LSTMs have the advantage of automatic feature learning and unbounded feature scope. Huang and Houfeng (2016) and Li et al. (2017) both employ bidirectional LSTM models to solve the grammatical error identification problem. To take advantage of the merits of both CRF and LSTM, there are also several participants who attempt to combine the two methods in order to achieve optimal results. Xie et al. (2017) propose a LSTM-CRF model trained on grammatical features such as character bigram, POS score, PMI score (Church and Hanks, 1990) and so on. Liao et al. (2017) also put forward a model that combines Bi-LSTM and CRF.
3 Task Data and Tools

In order to deal with the task of grammatical error identification for learners of Chinese as a foreign language, a number of experiments have been conducted on data from the above mentioned 2016 and 2017 shared tasks using the CRF model (Lafferty et al., 2001). During the preprocessing of the data, Stanford CoreNLP (Manning et al., 2014) is also used. The task data and tools will be described in depth in this chapter.

3.1 Task and Data

The task aims to tackle the problem of Chinese grammatical error diagnosis for learners of Chinese as a foreign language. Errors to be identified are R (redundant), M (missing), S (selection) and W (word order). One or more errors are contained in each input document which contains 1 to 5 sentences. Examples taken from the training data are displayed as follows.

<table>
<thead>
<tr>
<th>Example 1</th>
<th>Input:</th>
<th>Correction:</th>
<th>Error Interval:</th>
<th>Error Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>并且通过对话努力互相了解</td>
<td>并且通过对话努力互相了解</td>
<td>7, 7</td>
<td>R</td>
</tr>
<tr>
<td>Example 2</td>
<td>Input:</td>
<td>Correction:</td>
<td>Error Interval:</td>
<td>Error Type:</td>
</tr>
<tr>
<td></td>
<td>我同意这些现象</td>
<td>我支持这些现象</td>
<td>2, 3</td>
<td>S</td>
</tr>
<tr>
<td>Example 3</td>
<td>Input:</td>
<td>Correction:</td>
<td>Error Interval:</td>
<td>Error Type:</td>
</tr>
<tr>
<td></td>
<td>可是如果没有</td>
<td>可是如果没有</td>
<td>10, 10</td>
<td>M</td>
</tr>
<tr>
<td>Example 4</td>
<td>Input:</td>
<td>Correction:</td>
<td>Error Interval:</td>
<td>Error Type:</td>
</tr>
<tr>
<td></td>
<td>以前法律上没有</td>
<td>以前法例上没有</td>
<td>3, 8</td>
<td>W</td>
</tr>
</tbody>
</table>

Table 3.1: Examples from the training set

As can be seen from Table 3.1, there are four example documents which consist of input, correction, error interval and error type. In example 1, the error interval...
is 7, 7 and the error type is R, which means the 7th character ‘要’ in the input is redundant and therefore needs to be eliminated. In example 2, the error interval is 2, 3 and the error type is S, which means the 2nd and 3rd character ‘同意’ are selected inappropriately and therefore need to be replaced by ‘支持’. In example 3, the error interval is 10, 10 and the error type is M, which means the character ‘有’ should have been the 10th character in the input but instead is missing. In example 4, the error interval is 3, 8 and the error type is W, which means characters starting from the 3rd character ‘还’ to the 8th character ‘上’ are not in the correct sequence and should be corrected from ‘还没有法律上’ to ‘法律上还没有’.

The goal of the task is the same as that of the 2017 shared task, which is to develop a system which is able to identify whether a document is correct or not, which error types are present in the document and their exact locations. If a document does not contain any errors, the system should return the document number and the label correct. Otherwise, the system should return the document number, identified error type and a start point and an end point which indicate the position of the error. The results of the system will be evaluated on three level:

1. On the detection level, the system needs to classify all sentences as correct or incorrect. All error types are considered as incorrect.
2. On the identification level, the system is required to identify all error types in a certain sentence.
3. On the position level, the system has to produce the exact range at which the identified errors occur.

The evaluation metrics are in line with that of the 2016 and 2017 shared tasks.

<table>
<thead>
<tr>
<th>Confusion Matrix</th>
<th>System Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold Standard</td>
<td>Positive</td>
</tr>
<tr>
<td>Positive</td>
<td>TP (True Positive)</td>
</tr>
<tr>
<td>Negative</td>
<td>FP (False Positive)</td>
</tr>
</tbody>
</table>

Table 3.2: Confusion matrix for evaluation

As shown in the confusion matrix (Gaoqi et al., 2017), on the detection level, TP or true positive stands for the number of erroneous sentences that have been correctly identified as erroneous by the system. FP or false positive stands for the number of correct sentences that have been falsely identified as erroneous by the system. TN or true negative represents the number of correct sentences that have been correctly diagnosed as correct. FN or false negative represents the number of erroneous sentences that have been incorrectly diagnosed as correct. The evaluation metrics that can be calculated according to the confusion matrix are listed as follows:

- False Positive Rate = \( \frac{FP}{FP + TN} \)
- Accuracy = \( \frac{TP + TN}{TP + FP + TN + FN} \)
• $Precision = \frac{TP}{TP + FP}$

• $Recall = \frac{TP}{TP + FN}$

• $F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$

As for evaluation on the identification level and the position level, the evaluation metrics are calculated on individual error instance level instead of sentence level as on the detection level. Also, false positive rate is only calculated on the detection level. The organizers of the shared task provided a scoring script\(^1\) for the participants to evaluate their systems and it is also used as the evaluation tool in this thesis.

For instance, according to Gaoqi et al. (2017), the scoring script will generate the following scores given the four test inputs and corresponding system outputs displayed in Table 3.3.

<table>
<thead>
<tr>
<th>Gold Standard</th>
<th>System Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOC NO: 00038800481 6, 7, S 8, 8, R</td>
<td>DOC NO: 00038800481 2, 3, S 4, 5, S 8, 8, R</td>
</tr>
<tr>
<td>DOC NO: 00038800464 correct</td>
<td>DOC NO: 00038800464 correct</td>
</tr>
<tr>
<td>DOC NO: 00038801261 9, 9, M 16, 16, S</td>
<td>DOC NO: 00038801261 9, 9, M 16, 19, S</td>
</tr>
<tr>
<td>DOC NO: 00038801320 19, 25, W</td>
<td>DOC NO: 00038801320 19, 25, M</td>
</tr>
</tbody>
</table>

| Table 3.3: Gold standard and system output

• False Positive Rate = 0

• Detection Level
  - Accuracy = 4/4 = 1
  - Precision = 3/3 = 1
  - Recall = 3/3 = 1
  - $F1 = \frac{2 \times 1 \times 1}{1 + 1} = 1$

• Identification Level
  - Accuracy = 5/6 = 0.8333
  - Precision = 4/5 = 0.8
  - Recall = 4/5 = 0.8

\(^1\)http://ir.itec.nmtu.edu.tw/lre/nlpea16ged.htm.
F1 = 2*0.8*0.8/0.8+0.8 = 0.8

• Position Level
  - Accuracy = 3/7 = 0.4286
  - Precision = 2/6 = 0.3333
  - Recall = 2/5 = 0.4
  - F1 = 2*0.3333*0.4/0.3333+0.4 = 0.3636

The data sets used in this thesis come from both the 2016 shared task and the 2017 shared task. There are two tracks in the 2016 shared task: HSK track and TOCFL track. As for the 2017 shared task, only the HSK track is used. HSK stands for Hanyu ShuiPing Kaoshi, which means Test of Chinese Level (Cui and B.-l. Zhang, 2011; B. Zhang and Cui, 2013). TOCFL is the abbreviation of Test of Chinese as a Foreign Language (Lee, L.-P. Chang, and Tseng, 2016). HSK is in simplified Chinese while TOCFL is in traditional Chinese. Data used in this thesis is solely from the HSK track.

<table>
<thead>
<tr>
<th>Training Set</th>
<th>Development Set</th>
<th>Test Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017 training set</td>
<td>2017 test set</td>
<td>2016 test set</td>
</tr>
<tr>
<td>10426</td>
<td>3150</td>
<td>3011</td>
</tr>
</tbody>
</table>

Table 3.4: Data sets

As can be seen from Table 3.4, the training set used in this thesis is the training set from the 2017 shared task. As for the development set, we use the test set from the 2017 shared task. When it comes to the test set, test set from the 2016 shared task is used. The data sets used contain 10426, 3150 and 3011 input documents written by learners of Chinese as a foreign language respectively.

3.2 Conditional Random Field

The experiments in this thesis are conducted on the basis of the CRF model (Lafferty et al., 2001). This model has often been used to deal with sequence labeling problems and has been widely applied to many natural language processing tasks such as shallow parsing (Sha and Pereira, 2003), named entity recognition (Settles, 2004) and so on. CRF is a type of discriminative undirected graph model (Lafferty et al., 2001). Given observations X and random variables Y, in an undirected graph G = (V, E) in which V represents the vertices of G, Y = \{Y_v | v ∈ V\}, which means each node in V is connected to a random variable. (X,Y) is a conditional random field when the random variables which are conditioned on the given observations X conform to the Markov property regarding the graph:

P(Y_u|X, Y_{w \neq v}) = P(Y_u|X, Y_w, \neg v)

In the above formula, w~v means w and v are adjacent in G.

CRF can be used to tag sequential data. Based on the trained model, CRF can produce a corresponding label sequence Y for an input sequence X. In our
task, X represents the input sentence and each Y label is taken from the tag set \{R, M, S, W, O\}, which corresponds to redundant, missing, selection, word order error and correct usage respectively. The model can be denoted by the following formula:

\[
P(Y|X) = \frac{1}{Z(X)} \exp(\sum_k \lambda_k f_k)
\]

\(Z(X)\) corresponds to the normalization factor, \(f_k\) is the collection of features and \(\lambda_k\) represents the corresponding feature weights.

There are a number of available CRF-based tools such as CRFsuite\(^2\). CRFsuite supports first-order CRF and therefore is relatively fast. Since the script of our task is written in Python, python-crfsuite\(^3\), which is a python binding to CRFsuite, is chosen by us in this task.

### 3.3 Stanford CoreNLP

To achieve the final goal of identifying the grammatical errors in learner sentences, we need to first preprocess the data. Unlike English, no space exists to separate words from each other in Chinese. Therefore, we first need to perform segmentation. Since we plan to use POS tags and dependency labels as primary features, POS tagging and dependency parsing are also expected.

Stanford CoreNLP is an integrated NLP toolkit consisting of a wide range of grammatical analysis tools. In addition, it is also extremely easy to use and only requires approximately two lines of code on the command line to get the job done. Consequently, it perfectly fits our requirements in this task. The output of the Stanford CoreNLP is displayed in Figure 3.1.

As we can see from Figure 3.1, the second column contains the segmented words, the fourth column displays POS tags and the seventh column represents dependency labels. The POS tags are from the LDC Chinese Treebank\(^4\) POS tag set and the dependency labels are from the Universal Dependencies v1\(^5\).

---

\(^2\)http://www.chokkan.org/software/crfsuite/
\(^3\)https://github.com/scrapinghub/python-crfsuite
\(^4\)https://catalog.ldc.upenn.edu/LDC2013T21
\(^5\)http://universaldependencies.org/docsv1/
Figure 3.1: Output of Stanford CoreNLP

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>例如</td>
<td>AD</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>:</td>
<td>PU</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>3</td>
<td>青少年</td>
<td>NN</td>
<td>_</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>吸烟</td>
<td>VV</td>
<td>_</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>的</td>
<td>DEC</td>
<td>_</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>害处</td>
<td>NN</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>不仅</td>
<td>AD</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>是</td>
<td>VC</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>他</td>
<td>PN</td>
<td>_</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>本身</td>
<td>PN</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>11</td>
<td>还</td>
<td>AD</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>12</td>
<td>会</td>
<td>VV</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>13</td>
<td>对</td>
<td>P</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>14</td>
<td>社会</td>
<td>NN</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>15</td>
<td>的</td>
<td>DEG</td>
<td>_</td>
<td>14</td>
</tr>
<tr>
<td>16</td>
<td>未来</td>
<td>NT</td>
<td>_</td>
<td>17</td>
</tr>
<tr>
<td>17</td>
<td>发展</td>
<td>NN</td>
<td>_</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>。</td>
<td>PU</td>
<td>_</td>
<td>17</td>
</tr>
</tbody>
</table>
4 Experiments

We conduct a series of experiments to explore how we can improve the performance of the task of grammatical error identification for learners of Chinese as a foreign language. Specifically, we experiment with including dependency labels as additional features aside from features such as characters and POS tags, increasing the training set size and building separate classifiers for each error type. The details of these experiments will be discussed in this chapter.

4.1 Baseline System

The first step of building the baseline system is to prepare the dataset for training, which means we need to assign labels such as R (redundant), M (missing), W (word order), S (selection) and O (correct) to each character. Since the data is in XML format, the ElementTree XML api\(^1\) is used to parse the data. The result obtained is a list consisting of learner documents. Each document is also a list which contains tuples like (character, label). For example, to give a general idea of how the result looks like, we can take a look at part of the first element of the result:

```
[...('何', 'O'), ('而', 'S'), ('为', 'S'), ('而', 'R'), ('有', 'O')...]
```

We can then assign POS tags to each character and generate results with the similar structure but replacing the tuple (character, label) with (character, POS tag). One thing to note here is that Stanford CoreNLP can only produce POS tags for words instead of characters. Therefore, we use a BIO encoding (Kim et al., 2004) to assign POS tags to each character. Assume the POS tag of a certain word which contains one or more characters to be X, the BIO encoding will assign ‘B-X’ to the first character of the word and ‘I-X’ to the rest of the characters in the word. Having obtained the POS tagging results, we then go on to merge the two above mentioned results into the final result. The final result resembles the two previous results in structure but differ in that the tuple inside each document list has now become (character, POS tag, label).

With the final result, we can proceed to generate features to be used in the model training process. The features used in the baseline system are the character itself, its POS tag, the surrounding characters with a window size up to 6 and the POS tags of these characters. Python-crfsuite is used to train the CRF model. Firstly a trainer is created, then we feed the generated features and corresponding labels into the trainer. The next step is to set the parameters and call the train() function to initiate the training process. After the training process is completed, we can apply the trained model on our development or test data. What we

\(^1\)https://docs.python.org/3/library/xml.etree.elementtree.html
need to do next is to format the labeled development or test data so that its format is the same as that of the gold standard. The final step is to compare the predicted result with the gold standard using the provided evaluation tool. The baseline system manages to produce the following results after being evaluated. The evaluation results have been split into three tables corresponding to false positive rate, detection level, identification level and position level respectively.

<table>
<thead>
<tr>
<th>Detection Level</th>
<th>False Positive Rate</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0419</td>
<td>0.3886</td>
<td>0.6776</td>
<td>0.052</td>
<td>0.0966</td>
</tr>
<tr>
<td>2</td>
<td>0.0812</td>
<td>0.4054</td>
<td>0.6801</td>
<td>0.102</td>
<td>0.1774</td>
</tr>
<tr>
<td>3</td>
<td>0.1308</td>
<td>0.4149</td>
<td>0.6546</td>
<td>0.1465</td>
<td>0.2394</td>
</tr>
<tr>
<td>4</td>
<td>0.1769</td>
<td>0.4327</td>
<td>0.659</td>
<td>0.202</td>
<td>0.3092</td>
</tr>
<tr>
<td>5</td>
<td>0.2179</td>
<td>0.4403</td>
<td>0.6492</td>
<td>0.2384</td>
<td>0.3487</td>
</tr>
<tr>
<td>6</td>
<td>0.294</td>
<td>0.4559</td>
<td>0.6394</td>
<td>0.3081</td>
<td>0.4158</td>
</tr>
</tbody>
</table>

Table 4.1: False positive rate and detection level results based on window size 1~6 (dev set)

<table>
<thead>
<tr>
<th>Identification Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3779</td>
<td>0.4528</td>
<td>0.0217</td>
<td>0.0414</td>
</tr>
<tr>
<td>2</td>
<td>0.3856</td>
<td>0.4662</td>
<td>0.0437</td>
<td>0.08</td>
</tr>
<tr>
<td>3</td>
<td>0.3848</td>
<td>0.4357</td>
<td>0.0633</td>
<td>0.1106</td>
</tr>
<tr>
<td>4</td>
<td>0.3877</td>
<td>0.4204</td>
<td>0.0884</td>
<td>0.146</td>
</tr>
<tr>
<td>5</td>
<td>0.3838</td>
<td>0.4009</td>
<td>0.1055</td>
<td>0.1671</td>
</tr>
<tr>
<td>6</td>
<td>0.3846</td>
<td>0.4003</td>
<td>0.1429</td>
<td>0.2107</td>
</tr>
</tbody>
</table>

Table 4.2: Identification level results based on window size 1~6 (dev set)

<table>
<thead>
<tr>
<th>Position Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3616</td>
<td>0.1386</td>
<td>0.0047</td>
<td>0.0092</td>
</tr>
<tr>
<td>2</td>
<td>0.3495</td>
<td>0.1146</td>
<td>0.0078</td>
<td>0.0147</td>
</tr>
<tr>
<td>3</td>
<td>0.3302</td>
<td>0.0927</td>
<td>0.0099</td>
<td>0.0179</td>
</tr>
<tr>
<td>4</td>
<td>0.313</td>
<td>0.0942</td>
<td>0.0148</td>
<td>0.0257</td>
</tr>
<tr>
<td>5</td>
<td>0.2943</td>
<td>0.0857</td>
<td>0.0171</td>
<td>0.0285</td>
</tr>
<tr>
<td>6</td>
<td>0.2652</td>
<td>0.0825</td>
<td>0.0227</td>
<td>0.0356</td>
</tr>
</tbody>
</table>

Table 4.3: Position level results based on window size 1~6 (dev set)

We can see from the above tables that the false positive rate increases as the window size increases, which means that including more context in the training process leads to more correct sentences being falsely identified as erroneous by the system. On the detection level, it is shown that as window size increases, recall and F1 score goes up by a relatively large margin. Accuracy increases slightly.
while precision drops by a small margin. On the identification level, similar trends are observed. On the position level, accuracy and precision decreases with larger window size, while recall and F1-score increase slightly. Overall, the system performs best on the detection level, which makes sense because this is only a binary classification task. Since the identification level task is a multi-classification task, it produces worse results. The position level task produces the worst results, which indicates that this is the most difficult task among the three tasks. We assume the difficulty of predicting precisely the range in which the errors occur is due to the lack of delimiter between words in the Chinese language.

4.2 Modified Systems

4.2.1 Dealing with Overlapping Errors

After carefully examining the training data, we found that there was a problem in the training data which happened to be avoided in the baseline system. The problem is that there are some documents in which more than one error type is assigned to one character. For instance, if characters from index 1 to index 8 are assigned the error type ‘word disorder’ and characters from index 4 to 6 are assigned the error type ‘selection’, the characters from index 4 to 6 will be labeled as W and S at the same time. The baseline system manages to avoid this problem by overwriting the error type that is assigned earlier with the one that is assigned later. We have come up with another three approaches to work around this problem: the first approach is to delete all the documents that have this kind of problem. The second approach is to label the characters which are assigned more than one error type with an encoding like ‘W|S’ and later unpack the encoding in the process of evaluation. The third approach is to build separate classifiers to target each error type individually so that the problem of one character being assigned more than one error type can be solved.

1. In the deletion system, 515 documents have been deleted. According to the evaluation results, this system produces more or less the same or slightly worse results than the baseline system. However, it does outperform the baseline system in the detection level task. The results of the detection level task are as follows:

<table>
<thead>
<tr>
<th></th>
<th>Detection Level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
</tr>
<tr>
<td>1</td>
<td>0.3892</td>
</tr>
<tr>
<td>2</td>
<td>0.4067</td>
</tr>
<tr>
<td>3</td>
<td>0.419</td>
</tr>
<tr>
<td>4</td>
<td>0.4356</td>
</tr>
<tr>
<td>5</td>
<td>0.4425</td>
</tr>
<tr>
<td>6</td>
<td>0.4622</td>
</tr>
</tbody>
</table>

*Table 4.4: Detection level results for the deletion system (dev set)*
It can be seen from the table that after adopting the deletion system, accuracy, precision, recall and F1-score all increase by approximately one percent compared to the baseline system.

2. As for the encoding system, we observe that the running time is significantly longer than the baseline system and the deletion system. We believe it is probably due to the greatly increased number of labels from originally 5 labels to 21 labels after introducing encoding since the program is not able to distinguish the difference from ‘W|S’ and ‘S|W’ even though they are essentially the same encoding. Furthermore, the results produced by the encoding system unfortunately produces slightly worse recall and F1-score and better precision by only a tiny margin on each of the three levels compared to the baseline system.

3. When it comes to the separate classifier system, unlike the baseline system which only has one classifier for four error types, we have built four classifiers, each targeting one specific error type. For example, when the classifier for the ‘selection’ error type is used, each character will be labeled as either ‘S’ or ‘O’ depending on the annotation. Since each of the four classifiers will produce results of their own, we will have to merge the results produced by all of them in order to obtain the final results to be compared with the gold standard. During the process of merging, a certain document can only be labeled as ‘O’ if all of the four classifiers label it as ‘O’. It seems like a rather promising approach. However, to our surprise, it turns out that the results produced by this system are worse than the baseline system on almost every level. For example, the F1-score of both detection level and identification level drop by approximately one percent.

To conclude, after experimenting with three systems, it can be seen that only the deletion system manages to improve the results of the baseline system on the detection level. To put it in another way, an improved system can be built by integrating the detection level results of the deletion system and the results of the baseline system on the other two levels.

4.2.2 Increasing the Amount of Training Data

To further improve the system, we add the training data from the 2016 shared task which contains 9489 documents in order to double the training data to see if doing so will have a positive impact on the results. Nevertheless, according to the results, it is shown that even though doubling the training data does help decrease the false positive rate by almost 1.5 percent, the results on the detection level are worse than the improved baseline system. Specifically, for the detection level, precision increases slightly but recall decreases by almost 12 percent and therefore also hurts the F1-score. For the identification level, precision rises by almost 6 percent while the decrease of recall by approximately 6 percent also leads to a drop in F1-score in the end. A similar trend is found for the position level except recall and F1-score drop only by a small margin. To sum up, doubling the training data does not have a positive influence on the results as expected. We believe the reason for that might be that the 2016 and
2017 training data focus on different topics since the training data are taken from essays written by learners of Chinese as a foreign language. To put it in another way, the extra training data from a different domain might lead to the decrease in performance.

4.2.3 Adding Syntactic Features

Another experiment we have conducted is to add dependency labels as additional features. After obtaining dependency labels for each word using Stanford CoreNLP, a BIO encoding is used to assign dependency labels to each character. With dependency label as an additional feature, the false positive rate increases by approximately 2 percent. The results obtained on the other three levels are displayed in the following tables:

<table>
<thead>
<tr>
<th>Detection Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.394</td>
<td>0.6749</td>
<td>0.0692</td>
<td>0.1255</td>
</tr>
<tr>
<td>2</td>
<td>0.4095</td>
<td>0.6714</td>
<td>0.1187</td>
<td>0.2017</td>
</tr>
<tr>
<td>3</td>
<td>0.4165</td>
<td>0.6398</td>
<td>0.1641</td>
<td>0.2613</td>
</tr>
<tr>
<td>4</td>
<td>0.4448</td>
<td>0.6657</td>
<td>0.2343</td>
<td>0.3467</td>
</tr>
<tr>
<td>5</td>
<td>0.4594</td>
<td>0.6544</td>
<td>0.2965</td>
<td>0.4081</td>
</tr>
<tr>
<td>6</td>
<td>0.4781</td>
<td>0.6544</td>
<td>0.3596</td>
<td>0.4641</td>
</tr>
</tbody>
</table>

Table 4.5: Detection level results after adding dependency labels to the improved system (dev set)

<table>
<thead>
<tr>
<th>Identification Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3813</td>
<td>0.4873</td>
<td>0.029</td>
<td>0.0547</td>
</tr>
<tr>
<td>2</td>
<td>0.3867</td>
<td>0.4603</td>
<td>0.0507</td>
<td>0.0913</td>
</tr>
<tr>
<td>3</td>
<td>0.3874</td>
<td>0.4426</td>
<td>0.0709</td>
<td>0.1222</td>
</tr>
<tr>
<td>4</td>
<td>0.3848</td>
<td>0.4069</td>
<td>0.0923</td>
<td>0.1504</td>
</tr>
<tr>
<td>5</td>
<td>0.3899</td>
<td>0.4062</td>
<td>0.1261</td>
<td>0.1924</td>
</tr>
<tr>
<td>6</td>
<td>0.3895</td>
<td>0.3984</td>
<td>0.1613</td>
<td>0.2297</td>
</tr>
</tbody>
</table>

Table 4.6: Identification level results after adding dependency labels to the improved system (dev set)

Compared to the results produced by the previous best system, we can observe from Table 4.5, Table 4.6 and Table 4.7 that on the detection level, precision increases slightly while recall and F1-score rise noticeably by almost 4 percent. On the identification level, precision at window size 1 increase 3 percent while at window size 6, precision drops by a tiny margin. Both recall and F1-score increase by approximately 2 percent after adding dependency label as additional feature. When it comes to the position level, precision decreases by a tiny margin,
Table 4.7: Position level results after adding dependency labels to the improved system (dev set)

<table>
<thead>
<tr>
<th>Position Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3595</td>
<td>0.1463</td>
<td>0.0062</td>
<td>0.0119</td>
</tr>
<tr>
<td>2</td>
<td>0.3452</td>
<td>0.1134</td>
<td>0.0091</td>
<td>0.0168</td>
</tr>
<tr>
<td>3</td>
<td>0.3272</td>
<td>0.0951</td>
<td>0.0111</td>
<td>0.0199</td>
</tr>
<tr>
<td>4</td>
<td>0.3102</td>
<td>0.1037</td>
<td>0.0175</td>
<td>0.03</td>
</tr>
<tr>
<td>5</td>
<td>0.2839</td>
<td>0.0847</td>
<td>0.02</td>
<td>0.0324</td>
</tr>
<tr>
<td>6</td>
<td>0.2549</td>
<td>0.0765</td>
<td>0.0239</td>
<td>0.0364</td>
</tr>
</tbody>
</table>

and recall and F1-score get a tiny rise. In other words, adding dependency labels does not make much of a difference to the position level results. To summarize, recall and F1-score benefit from adding dependency labels as additional features.

4.3 Final Test Results

After experimenting with all the above mentioned systems, we finally select one optimal system to apply on our test data. This system has dependency labels as additional features. On the detection level, it uses the results from the deletion system, and for the other two levels, it makes use of the results from the original baseline system.

We decide to run the test data on the selected system and only look at the results at window size 1 and 6 since the best precision usually occur at window size 1 and best recall and F1-score normally appear at window size 6 based on previous observations. The obtained results are as follows:

Table 4.8: Detection level results obtained using the selected system (test set)

<table>
<thead>
<tr>
<th>Detection Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5061</td>
<td>0.454</td>
<td>0.0503</td>
<td>0.0905</td>
</tr>
<tr>
<td>6</td>
<td>0.4895</td>
<td>0.467</td>
<td>0.3125</td>
<td>0.3744</td>
</tr>
</tbody>
</table>

Table 4.9: Identification level results obtained using the selected system (test set)

<table>
<thead>
<tr>
<th>Identification Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5038</td>
<td>0.3708</td>
<td>0.0265</td>
<td>0.0494</td>
</tr>
<tr>
<td>6</td>
<td>0.4192</td>
<td>0.2808</td>
<td>0.1428</td>
<td>0.1893</td>
</tr>
<tr>
<td>Position Level</td>
<td>Accuracy</td>
<td>Precision</td>
<td>Recall</td>
<td>F1-score</td>
</tr>
<tr>
<td>----------------</td>
<td>----------</td>
<td>-----------</td>
<td>--------</td>
<td>----------</td>
</tr>
<tr>
<td>1</td>
<td>0.4865</td>
<td>0.0963</td>
<td>0.0049</td>
<td>0.0093</td>
</tr>
<tr>
<td>6</td>
<td>0.3225</td>
<td>0.0634</td>
<td>0.0246</td>
<td>0.0355</td>
</tr>
</tbody>
</table>

Table 4.10: Position level results obtained using the selected system (test set)

In the 2016 shared task, on the detection level, the best scores for accuracy, precision, recall and F1-score are 0.6659, 0.8746, 0.9755, 0.6628 respectively. On the Identification level, the best scores are 0.6849, 0.8821, 0.5447 and 0.5215. On the Position level, the best scores are 0.6477, 0.7144, 0.3697 and 0.3855. In the 2017 shared task, the best scores on the detection level are 0.6465, 0.7894, 1 and 0.7716. On the identification level, the best scores are 0.5513, 0.606, 0.6099 and 0.5188. On the position level, the best scores are 0.4121, 0.3663, 0.2941 and 0.2693. It can be seen that we still have a long way to go. However, it should also be noted that our results are actually not directly comparable to the results from the 2016 and 2017 shared tasks because most of the participants have employed large amount of external data in the training process.

4.4 Discussion

Comparing the experimental results displayed in the above tables to the original baseline system, it can be seen that on the detection level, precision decreases considerably by 22 percent at window size 1 and 17 percent at window size 6. Recall increases by a tiny margin and F1-score drops by 4 percent. On the identification level, precision decreases by 8 percent at window size 1 and 12 percent at window size 6. Recall almost stays the same and F1-score drops by approximately 2 percent. As for the position level, precision decreases by 4 percent at window size 1 and 2 percent at window size 6. Recall and F1-score are almost the same as the baseline system.

The results have clearly fallen short of our expectations. It can be observed that precision suffers the most, especially on the detection level. The unusually large extent to which the precision decreases leads to our hypothesis that the topic or domain of the training data may have a significant impact on the task of grammatical error diagnosis. In the previous experiments, when we add the 2016 training data to the 2017 training data to double the training data, it is also proved to have a negative influence on the result.

To further confirm our hypothesis, we have conducted yet another experiment which uses the same selected system but trained on the combined data of 2016 and 2017 training data. If our assumption is correct, then including the 2016 training data should produce better results since 2016 training data and test data share similar topics. The results are displayed in the following tables.

Based on the results in Table 4.11, Table 4.12 and Table 4.13, we can see that precision indeed increases significantly on all three levels, while the trend for recall and F1-score is similar to that of the previous experiment which doubles the training data. On the position level, recall and F1 even increase by approximately 1 percent. Therefore, we come to the conclusion that the topic or domain of
the training data has a significant impact on the system results, especially on precision on the detection level.

<table>
<thead>
<tr>
<th>Detection Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5191</td>
<td>0.6111</td>
<td>0.0448</td>
<td>0.0835</td>
</tr>
<tr>
<td>6</td>
<td>0.5115</td>
<td>0.5008</td>
<td>0.2065</td>
<td>0.2924</td>
</tr>
</tbody>
</table>

**Table 4.11:** Detection level results obtained using the selected system and 2016+17 data (test set)

<table>
<thead>
<tr>
<th>Identification Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5126</td>
<td>0.4414</td>
<td>0.0197</td>
<td>0.0376</td>
</tr>
<tr>
<td>6</td>
<td>0.4753</td>
<td>0.3472</td>
<td>0.0971</td>
<td>0.1517</td>
</tr>
</tbody>
</table>

**Table 4.12:** Identification level results obtained using the selected system and 2016+17 data (test set)

<table>
<thead>
<tr>
<th>Position Level</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5041</td>
<td>0.2193</td>
<td>0.0068</td>
<td>0.0131</td>
</tr>
<tr>
<td>6</td>
<td>0.4246</td>
<td>0.1472</td>
<td>0.0303</td>
<td>0.0503</td>
</tr>
</tbody>
</table>

**Table 4.13:** Position level results obtained using the selected system and 2016+17 data (test set)
5 Conclusion

5.1 Conclusion

We have focused on dealing with the task of identifying grammatical errors for learners of Chinese as a foreign language using the CRF model in this thesis. The task can be divided into three subtasks. The first subtask is binary classification, or in other words, to identify whether a sentence is correct or not. The second subtask is to diagnose the specific error types in the sentences. The last subtask is to find out the exact location of errors. Out of the three tasks, the third task is the most difficult one since there are no delimiters between Chinese words.

After preprocessing the data, we attempt to build a baseline system by using the character itself, its POS tag and neighboring characters and their POS tags to train the CRF model. We then find that there are some characters which are assigned more than one error type in the training data. To solve this problem, we have put forward three modified systems:

- **Deletion system:** all the problematic data have been deleted by us.

- **Encoding system:** if one character is assigned more than one error type, we will first label it as the combination of the errors such as ‘W|S’ and then unpack the encoding in the evaluation process.

- **Separate classifier system:** we have built four classifiers, each targeting one specific error type. The result of each classifier will then be merged as the final result.

Among these three systems, only the deletion system proves to improve the baseline system by approximately 1 percent on each evaluation metric on the detection level. The other two systems produce worse results instead. An improved baseline system has been built using the detection level results from the deletion system and the results for the other two levels from the original baseline system.

To further tune the system, we have introduced another two approaches. One is to double the training data, the other is to add dependency labels as additional features. According to the results, doubling the training data has a negative impact on the results due to a mismatch of the topics in the training and test data sets. Nevertheless, adding dependency labels as additional features proves to contribute to improving recall. Based on the experimental results from the modified systems, we have finally selected an optimal system to run our test data on.

However, the results fall short of our expectations. After conducting another experiment by adding the 2016 training data to the selected system, we have confirmed our assumption that the topic or domain of the training data can have a huge influence on the system results.
To sum up, the main findings of this thesis are: 1) filtering training data is effective in dealing with overlapping errors and leads to an improvement of results on the detection level, 2) adding syntactic features improves recall, 3) domain shifts greatly hurt results as a whole, especially precision.

5.2 Future Work

Since we have realized the importance of the domain of the training data, it is essential to collect a large amount of data covering a wide range of topics in order to achieve better results. Since annotated data are not always readily accessible, we can download data from websites that specializes in language learning and try to format it to be used for training.

Moreover, as is known to all, CRF relies heavily on feature engineering. In the future, we can experiment with neural network methods that are good at automatic feature learning such as LSTMs or the combination of CRF and neural network methods to obtain more satisfying results.
Bibliography


Huang, Shen and WANG Houfeng (2016). “Bi-LSTM Neural Networks for Chinese Grammatical Error Diagnosis”. In: Proceedings of the 3rd Workshop on Natural


Schneider, David and Kathleen F McCoy (1998). “Recognizing syntactic errors in the writing of second language learners”. In: Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics (ACL) and 17th


