Grammatical Error Detection and Correction using a Single Maximum Entropy Model*

Peilu Wang, Zhongye Jia and Hai Zhao†
Key Laboratory of Shanghai Education Commission for
Intelligent Interaction and Cognitive Engineering,
Center for Brain-Like Computing and Machine Intelligence
Department of Computer Science and Engineering, Shanghai Jiao Tong University
800 Dongchuan Road, Shanghai 200240, China
{plwang1990,jia.zhongye}@gmail.com, zhaohai@cs.sjtu.edu.cn

Abstract

This paper describes the system of Shanghai Jiao Tong Unvierity team in the CoNLL-2014 shared task. Error correction operations are encoded as a group of predefined labels and therefore the task is formulized as a multi-label classification task. For training, labels are obtained through a strict rule-based approach. For decoding, errors are detected and corrected according to the classification results. A single maximum entropy model is used for the classification implementation incorporated with an improved feature selection algorithm. Our system achieved precision of 29.83, recall of 5.16 and F_0.5 of 15.24 in the official evaluation.

1 Introduction

The task of CoNLL-2014 is grammatical error correction which consists of detecting and correcting the grammatical errors in English essays written by non-native speakers (Ng et al., 2014). The research of grammatical error correction can potentially help millions of people in the world who are learning English as foreign language. Although there have been many works on grammatical error correction, the current approaches mainly focus on very limited error types and the result is far from satisfactory.

The CoNLL-2014 shared task, compared with the previous Help Our Own (HOO) tasks (Dale et al., 2012) considering only determiner and preposition errors and the CoNLL-2013 shared task focusing on five major types of errors, requires to correct all 28 types of errors (Ng et al., 2014).

One traditional strategy is designing a system combined of a set of sub-models, where each submodel is specialized for a specific subtask, for example, correcting one type of errors. This strategy is computationally efficient and can adopt different favorable features for each subtask. Top ranked systems in CoNLL-2013 (Rozovskaya et al., 2013; Kao et al., 2013; Xing et al., 2013; Yoshimoto et al., 2013; Xiang et al., 2013) are based on this strategy. However, the division of the model relies on prior-knowledges and the designing of different features for each sub-model requires a large amount of manual works. This shortage is especially notable in CoNLL-2014 shared task, since the number of error types is much larger and the composition of errors is more complicated than before.

In contrast, we follow the work in (Jia et al., 2013a; Zhao et al., 2009a), integrating everything into one model. This integrated system holds a merit that a one-way feature selection benefits the whole system and no additional process is needed to deal with the conflict or error propagation of every sub-models. Here is a glance of this method: A set of more detailed error types are generated automatically from the original 28 types of errors. The detailed error type can be regarded as the label of a word, thus the task of grammatical error detection is transformed to a multi-label classification task using maximum entropy model (Berger et al., 1996; Zhao et al., 2013). A feature selection approach is introduced to get effective features from large amounts of feature candidates. Once errors are detected through word label classification, a rule-based method is used to make corrections according to their labels.

The rest of the paper is organized as follows. Section 2 describes the system architecture. Section 3 introduces the feature selection approach

*This work was partially supported by the National Natural Science Foundation of China (Grant No.60903119, Grant No.61170114, and Grant No.61272248), the National Basic Research Program of China (Grant No.2013CB329401), the Science and Technology Commission of Shanghai Municipality (Grant No.13511500200), and the European Union Seventh Framework Program (Grant No.247619).

†Corresponding author

Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pages 74–82, Baltimore, Maryland, 26-27 July 2014.

and the features we used. Experiments and results are presented in section 5, followed by conclusion.

2 System Architecture

In our approach, the grammatical error detection is regarded as a multi-label classification task. At first, each token in training corpus is assigned a label according to the golden annotation. The construction of labels is rule based using an extended version of Levenshtein edit distance algorithm which will be discussed in the following subsection. Each label maps an edit operation to do the correction, thus the generated labels are much more detailed than the originial 28 error types. Then, a maximum entropy (ME) model is adopted as the classifier. With the labeled data, the process of grammatical error correction is just applying the edit operation mapped by each label, which is basically the reverse of the labeling phase.

2.1 Data Labeling

In CoNLL-2014 shared task, there are 28 error types but they can not be used directly as class labels, since these types are too general that they can hardly be corrected by applying one rule-based edit. For example, the correction of *Vform* (verb form) error type includes all verb form inflections such as converting a verb to its infinitive form, gerund form, past form and past participle and so on. Previous works (Dahlmeier et al., 2012; Rozovskaya et al., 2012; Kochmar et al., 2012) manually decompose each error types to more detailed subtypes. For example, in (Dahlmeier et al., 2012), the determinater errors are decomposed into:

- replacement determiner (RD): { *a* → *the* }
- missing determiner (MD): { ϵ → *a* }
- unwanted determiner (UD): { *a* → ϵ }

For a task with a few error types such as merely determinative and preposition error in HOO 2012, manually decomposition may be sufficient. However, for CoNLL-2014, all 28 error types are required to be corrected and some of these types such as *Rloc-* (Local redundancy) and *Um* (Unclear meaning) are quite complex that the manual decomposition is time consuming and requires lots of grammatical knowledges. Therefore, an automatica decomposition method is proposed. It is extended from the Levenshtein edit distance algorithm and can divide error types into more detailed subtypes that each subtype can be corrected by applying one simple rule. How to calculate the extended Levenshtein edit distance is described in Algorithm 1.

Algorithm 1 Extended Levenshtein Edit Distance

INPUT: $toks_{src}, toks_{dst}$
OUTPUT: $\mathbb{E}, \mathbb{P}$
$l_{src}, l_{dst} \leftarrow \text{len}(toks_{src}), \text{len}(toks_{dst})$
$D[0 \ldots l_{src}][0 \ldots l_{dst}] \leftarrow 0$
$B[0 \ldots l_{src}][0 \ldots l_{dst}] \leftarrow (0, 0)$
$E[0 \ldots l_{src}][0 \ldots l_{dst}] \leftarrow \phi$
for $i \leftarrow 1 \ldots l_{src}$ **do**
  $D[i][0] \leftarrow i$
  $B[i][0] \leftarrow (i\text{-}1, 0)$
  $E[i][0] \leftarrow \mathcal{D}$
end for
for $j \leftarrow 1 \ldots l_{dst}$ **do**
  $D[0][j] \leftarrow j$
  $B[0][j] \leftarrow (0, j\text{-}1)$
  $E[0][j] \leftarrow \mathcal{A}$
end for
for $i \leftarrow 1 \ldots l_{src}$; **do**
  for $j \leftarrow 1 \ldots l_{dst}$ **do**
    if $toks_{src}[i\text{-}1] = toks_{dst}[j\text{-}1]$ **then**
      $D[i][j] \leftarrow D[i\text{-}1][j\text{-}1]$
      $B[i][j] \leftarrow (i\text{-}1, j\text{-}1)$
      $E[i][j] \leftarrow \mathcal{U}$
    else
      $m = \min(D[i\text{-}1][j\text{-}1], D[i\text{-}1][j], D[i][j\text{-}1])$
      if $m = D[i\text{-}1][j\text{-}1]$ **then**
        $D[i][j] \leftarrow D[i\text{-}1][j\text{-}1] + 1$
        $B[i][j] \leftarrow (i\text{-}1, j\text{-}1)$
        if $lemma(toks_{src}[i\text{-}1])$
          $= lemma(toks_{dst}[j\text{-}1])$ **then**
          $E[i][j] \leftarrow \mathcal{S}$
        else
          $E[i][j] \leftarrow \mathcal{I}$
        end if
      else if $m = D[i\text{-}1][j]$ **then**
        $D[i][j] \leftarrow D[i\text{-}1][j] + 1$
        $B[i][j] \leftarrow (i\text{-}1, j)$
        $E[i][j] \leftarrow \mathcal{D}$
      else if $m = D[i][j\text{-}1]$ **then**
        $D[i][j] \leftarrow D[i][j\text{-}1] + 1$
        $B[i][j] \leftarrow (i, j\text{-}1)$
        $E[i][j] \leftarrow \mathcal{A}$
      end if
    end if
  end for
end for
$i, j \leftarrow l_{src}, l_{dst}$
while $i > 0 \vee j > 0$ **do**
  insert $E[i][j]$ into head of $\mathbb{E}$
  insert $toks_{dst}[j-1]$ into head of $\mathbb{P}$
  $(i, j) \leftarrow B[i][j]$
end while
return $(\mathbb{E}, \mathbb{P})$

In this algorithm, $toks_{src}$ represents the tokens that are annotated with one grammatical error and $toks_{dst}$ represents the corrected tokens of $toks_{src}$. At first, three two dimensional matrixes D, B and

E are initialized. For all i and j, $D[i][j]$ holds the Levenshtein distance between the first i tokens of $toks_{src}$ and first j tokens of $toks_{dst}$. B stores the path of the Levenshtein distance and E stores the edit operations in this path. The original Levenshtein edit distance has 4 edit operations: unchange ($\mathcal{U}$), addition ($\mathcal{A}$), deletion ($\mathcal{D}$) and substitution ($\mathcal{S}$). We extend the "substitution" edit into two types of edits: inflection ($\mathcal{I}$) and the original substitution ($\mathcal{S}$). If two different words have the same lemma, the substitution operation is $\mathcal{I}$, else is $\mathcal{S}$. $lemma(x)$ returns the lemma of token x. This algorithm returns the edit operations $\mathbb{E}$ and the parameters of these operations $\mathbb{P}$. Here is a simple sample illustrating this algorithm. For the golden edit $\{$*a red apple is* $\rightarrow$ *red apples are*$\}$, $toks_{src}$ is *a red apple is*, $toks_{dst}$ is *red apples are*, the output edits $\mathbb{E}$ will be $\{\mathcal{D}, \mathcal{U}, \mathcal{I}, \mathcal{S}\}$, and the parameters $\mathbb{P}$ will be $\{$-, *red, apples, are*$\}$.

Then with the output of this extended Levenshtein distance algorithm, labels can be generated by transforming these edit operations into readable symbols. For those tokens without errors, we directly assign a special label "$\odot$" to them. A tricky part of the labeling process is the problem of the edit "addition", $\mathcal{A}$. A new token can only be added before or after an existing token. Thus for edit operation with addition, we must find an existing token that the label can be assigned to, and this sort of token is defined as *pivot*. A pivot can be a token that is not changed in an edit operation, such as the "*apple*" in edit $\{$*apple* $\rightarrow$ *an apple*$\}$, or some other types of edit such as the inflection of "*look*" to "*looking*" in edit $\{$*look* $\rightarrow$ *have been looking at*$\}$.

The names of these labels are based on BNF syntax which is defined in Figure 1. The non-terminal $\langle$*word*$\rangle$ can be substituted by all words in the vocabulary. The non-terminal $\langle$*inflection-rules*$\rangle$ can be substituted by terminals of inflection rules that are used for correcting the error types of noun number, verb form, and subject-verb agreement errors. All the inflection rules are listed in Table 1.

With the output of extended Levenshtein edits distance algorithm, Algorithm 2 gives the process to generate labels whose names are based on the syntax defined in Figure 1. It takes the output $\mathbb{E}$, $\mathbb{P}$ of Algorithm 1 as inputs and returns the generated set of labels $\mathbb{L}$. Each label in $\mathbb{L}$ corresponds to one token in $toks_{src}$ in order. For our previous example of edit $\{$*a red apple is* $\rightarrow$ *red apples are*$\}$, the $\mathbb{L}$ returned by Algorithm 2 is $\{\ominus$, $\odot$, NPLURAL, ARE$\}$ corresponding to the tokens $\{$*a*, *red*, *apple*, *is*$\}$ in $toks_{src}$. Some other examples of the generated labels are presented in Table 2.

⟨*label*⟩ ::= ⟨*simple-label*⟩ | ⟨*compound-label*⟩

⟨*simple-label*⟩ ::= ⟨*pivot*⟩ | ⟨*add-before*⟩ | ⟨*add-after*⟩

⟨*compound-label*⟩ ::= ⟨*add-before*⟩ ⟨*pivot*⟩
| ⟨*pivot*⟩ ⟨*add-after*⟩
| ⟨*add-before*⟩ ⟨*pivot*⟩ ⟨*add-after*⟩

⟨*pivot*⟩ ::= ⟨*unchange*⟩ | ⟨*substitution*⟩ | ⟨*inflection*⟩
| ⟨*deletion*⟩

⟨*add-before*⟩ ::= ⟨*word*⟩⊕
| ⟨*word*⟩⊕⟨*add-before*⟩

⟨*add-after*⟩ ::= ⊕⟨*word*⟩
| ⊕⟨*word*⟩⟨*add-after*⟩

⟨*substitution*⟩ ::= ⟨*word*⟩

⟨*inflection*⟩ ::= ⟨*inflection-rules*⟩

⟨*unchange*⟩ ::= ⊙

⟨*deletion*⟩ ::= ⊖

Figure 1: BNF syntax of label

Rules	Description
LEMMA	change word to its lemma
NPLURAL	change noun to its plural form
VSINGULAR	change verb to its singular form
GERUND	change verb to its gerund form
PAST	change verb to its past form
PART	change verb to its past participle

Table 1: Inflection rules

These labels are elaborately designed that each of them can be interpreted easily as a series of edit operations. Once the labels are determined by classifier, the correction of the grammatical errors is conducted by applying the edit operations interpreted from these labels.

Algorithm 2 Labeling Algorithm

INPUT: $\mathbb{E}, \mathbb{P}$
OUTPUT: $\mathbb{L}$
$pivot \leftarrow$ number of edits in $\mathbb{E}$ that are not $\mathcal{A}$
$\mathbb{L} \leftarrow \phi$
$L \leftarrow ''$
while $i <$ length($\mathbb{E}$) **do**
  if $\mathbb{E}[i] = \mathcal{A}$ **then**
    $L \leftarrow L+$ label of edit $\mathbb{E}[i]$ with $\mathbb{P}[i]$
    $i \leftarrow i + 1$
  else
    $l \leftarrow L+$ label of edit $\mathbb{E}[i]$ with $\mathbb{P}[i]$
    $pivot \leftarrow pivot - 1$
    if $pivot = 0$ **then**
      $i \leftarrow i + 1$
      while $i <$ length of $\mathbb{E}$ **do**
        $l \leftarrow l + \oplus + \mathbb{P}[i]$
        $i \leftarrow i + 1$
      end while
    end if
    push l into $\mathbb{L}$
    $L \leftarrow ''$
  end if
end while
$\mathbb{L} \leftarrow$ upper case of $\mathbb{L}$
return $\mathbb{L}$

Tokens	Edit	Label
to reveal	{*to reveal→revealing*}	$\ominus$ GERUND
a woman	{*a woman→women*}	$\ominus$ NPLURAL
developing wold	{*developing world →the developing world*}	THE$\oplus$ $\odot$
a	{*a*→ ϵ}	$\ominus$
in	{*in→on*}	ON
apple	{*apple→an apple*}	AN$\oplus$

Table 2: Examples of labeling

2.2 Label Classification

Using the approach described above, the training corpus is converted to a sequence of words with labels. Maximum entropy model is used as the classifier. It allows a very rich set of features to be used in a model and has shown good performance in similiar tasks (Zhao et al., 2013). The features we used are discussed in the next section.

3 Feature Selection and Generation

One key factor affecting the performance of maximum entropy classifier is the features it used. A good feature that contains useful information to guide classification will significantly improve the performance of the classifier. One direct way to involve more good features is involving more features.

In our approach, large amounts of candidate features are collected at first. We carefully examine the factors involved in a wide range of features that have been or can be used to the word label classification task. Many features that are considered effective in various of previous works (Dahlmeier et al., 2012; Rozovskaya et al., 2012; Han et al., 2006; Rozovskaya et al., 2011; Tetreault, Joel R and Chodorow, Martin, 2008) are included. Besides, features that are used in the similar spell checking tasks (Jia et al., 2013b; Yang et al., 2012) and some novel features showing effectiveness in other NLP tasks (Wang et al., 2013; Zhang and Zhao, 2013; Xu and Zhao, 2012; Ma and Zhao, 2012; Zhao, 2009; Zhao et al., 2009b) are also included. However, using too many features is time consuming. Besides, it increases the probability of overfitting and may lead to a poor solution of the maximum-likelihood parameter estimate in the ME training.

Algorithm 3 Greedy Feature Selection

INPUT: all feature candidates F
OUTPUT: selected features S
$S = \{f_0, f_1, \ldots, f_k\}$, a random subset of F
while do
  $C =$ RECRUITMORE(S)
  if $C = \{\}$ **then**
    return S
  end if
  $S' =$ SHAKEOFF(S+C)
  if $scr(M(S)) \geq scr(M(S'))$ **then**
    return S
  end if
  $S = S'$
end while
function RECRUITMORE(S)
  $C = \{\}$, and $p = scr(M(S))$
  for each $f \in F - S$ **do**
    if $p < scr(M(S + \{f\}))$ **then**
      $C = C + \{f\}$
    end if
  end for
end function
function SHAKEOFF(S)
  while do
    $S' = S_0 = S$
    for each $f \in S$ **do**
      if $scr(M(S')) < scr(M(S' - \{f\}))$ **then**
        $S' = S' - \{f\}$
      end if
    end for
    $S = S'$
    if $S' = S_0$ **then**
      return S'
    end if
  end while
end function

Therefore a feature selection algorithm is introduced to filter out "bad" features at first and the remaining features will be used to generate new features. The feature selection algorithm has shown effectiveness in (Zhao et al., 2013) and is presented in Algorithm 3.

In this algorithm, $M(S)$ represents the model using feature set S and $scr(M)$ represents the evaluation score of model M on a development data set. It repeats two main steps until no further performance gain is achievable:

1. Include any features from the rest of F into the current set of candidate features if the inclusion would lead to a performance gain.

2. Exclude any features from the current set of candidate templates if the exclusion would lead to no deterioration in performance.

By repeatedly adding the useful and removing the useless features, the algorithm aims to return a better and smaller set of features for next round. Only 55 of the 109 candidate features remain after using this algorithm and they are presented in Table 4. Table 3 gives an interpretation of the abbreviations used in Table 4. Each feature of a word is set to that listed in **feature** column if the word satisfies the condition listed in **current word** column, else the feature is set to "NULL". For example, if the current word satisfies the condition in the first row of Table 4 which is the first word in the left of a *NC*, feature 1 of this word is set to all words in the *NC*, otherwise, feature 1 is set to "NULL".

Abbreviation	Description
NP	*Noun Phrase*
NC	*Noun Compound* and is active if second to last word in *NP* is tagged as noun
VP	*Verb Phrase*
cw	*Current Word*
pos	part-of-speech of the current word
$X.l_i$	the ith word in the left of *X*
$X.r_i$	the ith word in the right of *X*
NP[0]	the first word of *NP*
NP.head	the head word of *NP*
NP.(DT or IN or TO)	word in *NP* whose pos is DT or IN or TO
VP.verb	word in *VP* whose pos is verb
VP.NP	*NP* in *VP*
dp	the dependency relation generated by standford dependency parser
dp.dep	the dependent in the dependency relation
dp.head	the head in the dependency relation
dp.rel	the type of the dependency relation

Table 3: The interpretation of the abbrevations in Table 4

4 Experiment

4.1 Data Sets

The CoNLL-2014 training data is a corpus of learner English provided by (Dahlmeier et al., 2013). This corpus consists of 1,397 articles, 12K sentences and 116K tokens. The official blind test data consists of 50 articles, 245 sentences and 30K tokens. More detailed information about this data is described in (Ng et al., 2014; Dahlmeier et al., 2013).

In development phase, the entire training corpus is splited by sentence. 80% sentences are picked up randomly and used for training and the rest 20% are used as the developing corpus. For the final submission, the entire corpus is used for training.

4.2 Data Labeling

The labeling algorithm described in section 2.1 is firstly applied to the training corpus. Total 7047 labels are generated and those whose count is larger than 15 is presented in Table 5. Directly applying these 7047 labels for correction receives an M^2 score of precision=90.2%, recall=87.0% and F_0.5=89.5%. However, the number of labels is too large that the training process is time consuming and those labels appears only few times will hurt the generalization of the trained model. Therefore, labels with low frequency which appear less than 30 times are cut out and 109 labels remain. The M^2 score of the system using this refined labels is precision=83.9%, recall=64.0% and F_0.5=79.0%. Note that even applying all labels, the F_0.5 is not 100%. It is because some annotations in the training corpus are not consistency.

current word	feature
$NC.l_1$	NC
$NP.l_1$	NP
$NP[0]$	$NP.l_1$.pos
$NC.l_1$	NC
$NC.l_1$	$NC.l_1$.pos
$NC.l_1$ and pos=DT	NC
$NC.l_1$ and pos=VB	NC
$NP.l_1$ and pos=VB	NP
pos=VB	cw
pos=DT	cw
the	$cw.r_1$
a	$cw.r_1$
an	$cw.r_1$
$NP[0]$	cw
$NP[0]$	$NP.l_1$
$NP[0]$	$NP.l_2$
$NP[0]$	$NP.l_3$
$NP[0]$	$NP.l_1$.pos
$NP[0]$	$NP.l_2$.pos
$NP[0]$	$NP.l_3$.pos
$NP.l_1$	NP.head
$NP.l_1$	NP.head.pos
NP.head	NP. head
NP.head	NP. head.bag
NP.head	NP. head.pos
NP.head	NP. head.pos.bag
NP.head	NP. (JJ or CC)
NP.(DT or IN or TO)	NP
NP.(DT or IN or TO)	NP.head
NP.(DT or IN or TO)	NP.head.pos
dp.dep	dp.head
dp.head	dp.dep
dp.dep	dp.head.pos
dp.head	dp.dep.pos
dp.dep	dp.rel
dp.head	dp.rel
VP.verb	$VP.NP$
VP.verb	$VP.NP$.head
$VP.NP$.head	VP.verb
VP.verb	$VP.NP$.head.pos
$VP.NP$.head	VP.verb.pos
cw	$cw.l_i,\ i \in \{0,1,2,3\}$
cw	$cw.r_i,\ i \in \{1,2,3\}$
cw	$cw.l_i$.pos, $i \in \{0,1,2,3\}$
cw	$cw.r_i$.pos, $i \in \{1,2,3\}$

Table 4: Remained features after the feature selection.

Count	Label
1091911	⊙
31507	⊖
3637	NPLURAL
2822	THE⊕
2600	LEMMA
948	,⊕
300~900	A⊕ PAST THE IN TO . IS OF ARE FOR GERUND ,
50~100	AND ON AN⊕ A VSINGULAR WAS THEIR
20~50	ELDERLY IT OF⊕ THEY WITH TO⊕ WERE THIS ; ITS .⊕ THAT 'S⊕ AND⊕ THAT⊕ HAVE⊕ CAN AS HAVE⊕PART FROM BE WOULD BY
15~20	HAVE HAS⊕ WILL HAS AT AN THESE ⊕, THEM IN⊕ INTO #⊕ ARE⊕ WHICH PEOPLE HAS⊕PART ECONOMIC IS⊕ BE⊕ SO COULD TO⊕LEMMA MANY PART MAY LESS IT⊕ FOR⊕ BEING⊕
15~20	NOT ABOUT WILL⊕LEMMA SHOULD HIS BECAUSE AGED SUCH ALSO WHICH⊕ HAVE⊕PAST WILL⊕ WHO WHEN MUCH
15~20	ON⊕ ' THROUGH BE⊕PAST MORE IF HELP THE⊕ELDERLY 'S ONE AS⊕ THERE THEIR⊕ WITH⊕ HAVE⊕⊙ ECONOMY DEVELOPMENT CONCERNED PEOPLE⊕ PROBLEMS BUT MEANS THEREFORE HOWEVER BEING : UP PROBLEM '⊕ THE⊕LEMMA IN⊕ADDITION HOWEVER⊕,⊕ AMONG ;⊕ WHERE THUS ONLY HEALTH HAS⊕PAST FUNDING EXTENT ALSO⊕ TECHNOLOGICAL " OR HAD WOULD⊕ VERY .⊕THIS ITS⊕ IMPORTANT DEVELOPED ⊕BEEN AGE ABOUT⊕ WHO⊕ USE THEY⊕ THAN NUMBER HOWEVER⊕, GOVERNMENT FURTHERMORE DURING BUT⊕ YOUNGER RIGHT POPULATION PERSON⊕ FEWER ENVIRONMENTALLY WOULD⊕LEMMA OTHER MAY⊕ LIMITED HE COULD⊕HAVE BEEN STILL SPENDING SAFETY OVER ONE⊕'S MAKE MADE LIFE HUMAN HAD⊕ FUNDS CARE ARGUED ALL "⊕ WHEN⊕ TIME THOSE SOCIETY RESEARCH PROVIDE OLD NEEDS INCREASING DEVELOPING BECOME BE⊕⊙ ADDITION

Table 5: Labels whose count is larger than 15.

current word	feature
$NC.l_1$	NC, cw, $cw.l_1$, $cw.l_1$.pos, $cw.r_1$, $cw.r_1$.pos
$NP[0]$	NP.head, $NP.l_1$, $NP.l_2$, cw, $cw.l_1$, $cw.l_1$.pos,
NP.head	$NP[0]$, $NP.l_1$, $NP.l_2$, cw, $cw.l_1$, $cw.l_1$.pos,
dp.head	cw, $cw.l_1$, $cw.l_2$ dp.dep, dp.dep.pos, dp.rel

Table 6: Examples of the new generated features.

4.3 Data Refinement

The training corpus is refined before used that sentences which do not contain errors are filtered out. Only 38% of the total sentences remain. With less training corpus, it takes less time to train the ME model. Table 7 presents the performance of systems using the unrefined training corpus and refined corpus.

System	Presicion	Recall	F_0.5
unrefined	26.99%	1.67%	6.71%
refined	11.17%	3.1%	7.34%

Table 7: Comparison of systems with different training corpus.

All sets of these systems are kept the same except the training corpus they use. It can be seen that the refinement also improves the performance of the system.

4.4 Feature Selection

Figure 2 shows the results of systems with different feature sets. *sys_10* is the system with

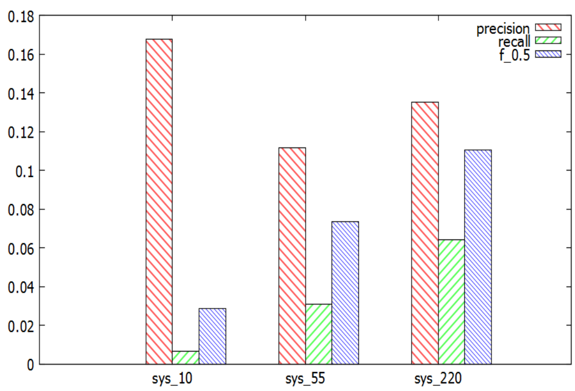

Figure 2: Performance of systems with different features.

10 randomly chosen features which are used as the initial set of features in Algorithm 3, *sys_55* is the system with the refined 55 features. With these refined features, various of new features are generated by combining different features. This combination is conducted empirically that features which are considered having relations are combined to generate new features. Using this method, 165 new features are generated and total 220 features are used in *sys_220*. Table 6 gives a few of examples showing the combined features. The performance is evaluated by the precision, recall and F_0.5 score of the M^2 scorer according to (Dahlmeier and Ng, 2012). It can be seen that *sys_220* with the most number of features achieves the best performance.

4.5 Evaluation Result

The final system we use is *sys_220* with refined training data, the performance of our system on the developing corpus and the blind official test data is presented in Table 8. The score is calculated using M^2 scorer.

Data Set	Precision	Recall	F_0.5
DEV	13.52%	6.41%	11.07%
OFFICIAL	29.83%	5.16%	15.24%

Table 8: Evaluation Results

5 Conclusion

In this paper, we describe the system of Shanghai Jiao Tong Univerity team in the CoNLL-2014 shared task. The grammatical error detection is regarded as a multi-label classification task and the correction is conducted by applying a rule-based approach based on these labels. A single maximum entropy classifier is introduced to do the multi-label classification. Various features are involved and a feature selection algorithm is used to refine these features. Finally, large amounts of feature templates that are generated by the combination of the refined features are used. This system achieved precision of 29.83%, recall of 5.16% and F_0.5 of 15.24% in the official evaluation.

References

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2012)*, pages 568–572, Montreal, Canada.

Daniel Dahlmeier, Hwee Tou Ng, and Eric Jun Feng Ng. 2012. NUS at the HOO 2012 Shared Task. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 216–224, Montréal, Canada, June. Association for Computational Linguistics.

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

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. Hoo 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, pages 54–62, Montréal, Canada, June. Association for Computational Linguistics.

NA-RAE Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting Errors in English Article Usage by Non-Native Speakers. *Natural Language Engineering*, 12:115–129, 5.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013a. Grammatical error correction as multiclass classification with single model. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 74–81, Sofia, Bulgaria, August. Association for Computational Linguistics.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013b. Graph model for chinese spell checking. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 88–92, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Ting-hui Kao, Yu-wei Chang, Hsun-wen Chiu, Tzu-Hsi Yen, Joanne Boisson, Jian-cheng Wu, and Jason S. Chang. 2013. Conll-2013 shared task: Grammatical error correction nthu system description. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 20–25, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ekaterina Kochmar, Øistein Andersen, and Ted Briscoe. 2012. HOO 2012 Error Recognition and Correction Shared Task: Cambridge University Submission Report. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 242–250, Montréal, Canada, June. Association for Computational Linguistics.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India, December. The COLING 2012 Organizing Committee.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014 Shared Task)*, Baltimore, Maryland, USA.

Alla Rozovskaya, Mark Sammons, Joshua Gioja, and Dan Roth. 2011. University of Illinois System in HOO Text Correction Shared Task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 263–266. Association for Computational Linguistics.

Alla Rozovskaya, Mark Sammons, and Dan Roth. 2012. The UI System in the HOO 2012 Shared Task on Error Correction. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 272–280, Montréal, Canada, June. Association for Computational Linguistics.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. 2013. The university of illinois system in the conll-2013 shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 13–19, Sofia, Bulgaria, August. Association for Computational Linguistics.

Tetreault, Joel R and Chodorow, Martin. 2008. The Ups and Downs of Preposition Error Detection in ESL Writing. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 865–872. Association for Computational Linguistics.

Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao, and Bao-Liang Lu. 2013. Converting continuous-space language models into n-gram language models for statistical machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 845–850, Seattle, Washington, USA, October. Association for Computational Linguistics.

Yang Xiang, Bo Yuan, Yaoyun Zhang, Xiaolong Wang, Wen Zheng, and Chongqiang Wei. 2013. A hybrid model for grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 115–122, Sofia, Bulgaria, August. Association for Computational Linguistics.

Junwen Xing, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Xiaodong Zeng. 2013. Um-checker: A hybrid system for english grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 34–42, Sofia, Bulgaria, August. Association for Computational Linguistics.

Qiongkai Xu and Hai Zhao. 2012. Using deep linguistic features for finding deceptive opinion spam. In *Proceedings of COLING 2012: Posters*, pages 1341–1350, Mumbai, India, December. The COLING 2012 Organizing Committee.

Shaohua Yang, Hai Zhao, Xiaolin Wang, and Bao liang Lu. 2012. Spell checking for chinese. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and

Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 730–736, Istanbul, Turkey, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L12-1423.

Ippei Yoshimoto, Tomoya Kose, Kensuke Mitsuzawa, Keisuke Sakaguchi, Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2013. Naist at 2013 conll grammatical error correction shared task. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 26–33, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jingyi Zhang and Hai Zhao. 2013. Improving function word alignment with frequency and syntactic information. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2211–2217. AAAI Press, August.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009a. Semantic dependency parsing of nombank and propbank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 30–39, Singapore, August. Association for Computational Linguistics.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009b. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 55–63, Suntec, Singapore, August. Association for Computational Linguistics.

Hai Zhao, Xiaotian Zhang, and Chunyu Kit. 2013. Integrative semantic dependency parsing via efficient large-scale feature selection. *Journal of Artificial Intelligence Research*, 46:203–233.

Hai Zhao. 2009. Character-level dependencies in chinese: Usefulness and learning. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 879–887, Athens, Greece, March. Association for Computational Linguistics.