NUS at the HOO 2012 Shared Task

Daniel Dahlmeier[1], **Hwee Tou Ng**[1,2], and **Eric Jun Feng Ng**[2]
[1]NUS Graduate School for Integrative Sciences and Engineering
[2]Department of Computer Science, National University of Singapore
{danielhe,nght,eng}@comp.nus.edu.sg

Abstract

This paper describes the submission of the National University of Singapore (NUS) to the HOO 2012 shared task. Our system uses a pipeline of confidence-weighted linear classifiers to correct determiner and preposition errors. Our system achieves the highest correction F_1 score on the official test set among all 14 participating teams, based on gold-standard edits both before and after revision.

1 Introduction

Grammatical error correction is the task of automatically detecting and correcting erroneous word usage and ill-formed grammatical constructions in text. Determiner and preposition errors are the two most prominent types of errors made by non-native speakers of English. Although there has been much work on automatic correction of determiner and preposition errors over the last few years, it has so far been impossible to directly compare results because different teams have evaluated on different data sets.

The HOO 2012 shared task evaluates grammatical error correction systems for determiner and preposition errors. Participants are provided with a set of documents written by non-native speakers of English. The task is to automatically detect and correct determiner and preposition errors and produce a set of corrections (called *edits*). Evaluation is done by computing precision, recall, and F_1 score between the system edits and a manually created set of gold-standard edits. The details of the HOO 2012 shared task are described in the official overview paper (Dale et al., 2012).

In this paper, we describe the system submission from the National University of Singapore (NUS). Our system treats determiner and preposition correction as classification problems. We use confidence-weighted linear classifiers to predict the correct word from a confusion set of possible correction options. Separate classifiers are built for determiner errors, preposition replacement errors, and preposition insertion and deletion errors. The classifiers are combined into a pipeline of correction steps to form an end-to-end error correction system. Our system achieves the highest correction F_1 score on the official test set among all 14 participating teams, based on gold-standard edits both before and after revision.

The remainder of this paper is organized as follows. The next section presents our error correction system. Section 3 describes the features. Section 4 presents experimental results. Section 5 contains further discussion. Section 6 concludes the paper.

2 System Architecture

Our system consists of a pipeline of sequential steps where the output of one step serves as the input to the next step. The steps in sequence are:

1. Pre-processing
2. Determiner correction (Det)
3. Replacement preposition correction (RT)
4. Missing and unwanted preposition correction (MT, UT)

The final output after the last step forms our submission to the shared task. Each correction step (i.e., steps 2, 3, 4) involves three internal steps:

1. Feature extraction

The 7th Workshop on the Innovative Use of NLP for Building Educational Applications, pages 216–224,
Montréal, Canada, June 3-8, 2012.

2. Classification
3. Language model filter

Feature extraction first analyzes the syntactic structure of the input sentences (part-of-speech (POS) tagging, chunking, and parsing) and identifies relevant instances for correction (e.g., all noun phrases (NP) for determiner correction). Each instance is mapped to a real-valued feature vector. Next, a classifier predicts the most likely correction for each feature vector. Finally, the proposed corrections are filtered using a language model and only corrections that strictly increase the language model score are kept.

2.1 Confidence-Weighted Learning

As the learning algorithm for all classifiers, we choose confidence-weighted (CW) learning (Dredze et al., 2008; Crammer et al., 2009), which has been shown to perform well for natural language processing (NLP) problems with high dimensional and sparse feature spaces. Instead of keeping a single weight vector, CW learning maintains a distribution over weight vectors, parametrized by a multivariate normal distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with mean $\boldsymbol{\mu}$ and covariance matrix Σ. In practice, Σ is often approximated by a diagonal matrix (Dredze et al., 2008). CW is an online learning algorithm that proceeds in rounds over a labeled training set $((y_1, \mathbf{x}_1), (y_2, \mathbf{x}_2), \ldots, (y_n, \mathbf{x}_n))$, one example at a time. After the i-th round, CW learning updates the distribution over weight vectors such that the i-th example is predicted correctly with probability at least $0 < \eta < 1$ while choosing the update step that minimizes the Kullback-Leibler (KL) distance from the current distribution. The CW update rule is:

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

Dredze *et al.* (2008) show that in the binary case, the CW update rule has a closed-form solution. In the multi-class case, there exists no closed-form solution but the solution can be efficiently approximated.

2.2 Pre-processing

Pre-processing involves sentence splitting, tokenization, re-casing, and spelling correction. We noticed that the HOO 2012 training data contained a large number of spelling mistakes and that some documents are written in all upper case. Both have a negative effect on tagging and classification accuracy. We automatically identify and re-case upper-case documents using a standard re-casing model from statistical machine translation (SMT). Re-casing is modeled as monotone decoding (without reordering) involving translation of an un-cased sentence to a mixed-case sentence. Next, we automatically correct spelling mistakes using an open-source spell checker. Words are excluded from spelling correction if they are shorter than a threshold (set to 4 characters in our work), or if they include hyphens or upper case characters inside the word. We apply a language model filter (described in the next subsection) to filter the proposed spelling corrections. Note that spelling correction is only performed to improve the accuracy of subsequent correction steps. Spelling corrections themselves are *not* part of the edits submitted for evaluation.

2.3 Determiner Correction

Determiner errors include three error types: replacement determiner (RD), missing determiner (MD), and unwanted determiner (UD). Although determiners are not limited to articles (*a, an, the, empty article* ϵ), article errors account for the majority of determiner errors. We therefore focus our efforts on errors involving only articles.

2.3.1 Correction as Classification

We treat determiner error correction as a multi-class classification problem. A classifier is trained to predict the correct article from a *confusion set* of possible article choices {*a*, *the*, ϵ}, given the sentence context. The article *an* is normalized as *a* and restored later using a rule-based heuristic. During training, every NP in the training data generates one training example. The class $y \in$ {*a*, *the*, ϵ} is the correct article as annotated by the gold standard or the observed article used by the writer if the article is not annotated (i.e., the article is correct). The surrounding context is represented as a real-valued feature vector $\mathbf{x} \in \mathcal{X}$. The features of our classifiers are described in Section 3.

One challenge in training classifiers for grammatical error correction is that the data is highly skewed.

Training examples without any error (i.e., the observed article equals the correct article) greatly outnumber those examples with an error (i.e., the observed article is different from the correct article). As the observed article is highly correlated with the correct article, the observed article is a valuable feature (Rozovskaya and Roth, 2010; Dahlmeier and Ng, 2011). However, the high correlation can have the undesirable effect that the classifier *always* predicts the observed article and never proposes any corrections. To mitigate this problem, we re-sample the training data, either by oversampling examples with an error or undersampling examples without an error. The sampling parameter is chosen through a grid search so as to maximize the F_1 score on the development data. After training, the classifier can be used to predict the correct article for NPs from new unseen sentences.

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

2.3.2 Language Model Filter

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

$$score_{lm} = \frac{1}{|\mathbf{s}|} \log Pr(\mathbf{s}), \qquad (2)$$

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

2.4 Replacement Preposition Correction

Replacement preposition correction follows the same strategy as determiner correction, but with a different confusion set and different features. The confusion set consists of 36 frequent prepositions which we adopt from our previous work (Dahlmeier and Ng, 2011).[1] These prepositions account for the majority of preposition replacement errors in the HOO 2012 training data. During training, every prepositional phrase (PP) in the training data which is headed by a preposition from the confusion set generates one training example. The class y is the correct preposition. During testing, every PP in the test data which is headed by a preposition from the confusion set generates one test example.

2.5 Missing Preposition Correction

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

We treat missing preposition (MT) correction as a binary classification problem.[2] For each preposition p, we train a binary classifier that predicts the presence or absence of that preposition. Thus, the confusion set consists only of the preposition p and the "empty preposition". During training, we require examples of contexts where p should be used and where it should be omitted. As prepositions typically appear before NPs, we take every NP in the training data as one training example. If the preposition p appears right in front of the NP (i.e., the preposition p and the NP form a PP), the example is a positive example, otherwise (i.e., another preposition or no preposition appears before the NP) it is a negative example. During testing, every NP which does not directly follow a preposition generates one test example. If the classifier predicts that the preposition p should have been used in this context with sufficiently high confidence and inserting p increases the normalized language model score, p is inserted before the NP.

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

[2] Alternatively, missing preposition error correction could be treated as a multi-class problem, but we found that binary classifiers gave better performance in initial experiments.

2.6 Unwanted Preposition Correction

Unwanted preposition correction is treated as a binary classification problem similar to missing preposition correction but with different training and test examples. When training the classifier for preposition p, every PP where the writer used the preposition p is one training example. If the gold-standard annotation labels p as unwanted, the example is a positive example for deleting p, otherwise it is a negative example. During testing, every PP with the preposition p generates one test example. If the classifier predicts that p should be deleted with sufficiently high confidence and deleting p increases the normalized language model score, p is deleted.

We found that separate classifiers for missing and unwanted preposition correction gave slightly better results compared to using a single classifier for both tasks. As the test examples for missing and unwanted preposition correction of a preposition p are disjoint, both steps can be performed in parallel. This also prevents the case of the system "contradicting" itself by first inserting a preposition and later deleting it. We perform missing preposition correction and unwanted preposition correction for each preposition in turn, before moving to the next preposition.

3 Features

In this section, we describe the features used in our system. The choice of features can have an important effect on classification performance. The exact features used for determiner, replacement preposition, and missing and unwanted preposition correction are listed in Tables 1, 2, 3, and 4, respectively. The features were chosen empirically through experiments on the development data.

The most commonly used features for grammatical error correction are lexical and POS N-grams, and chunk features. We adopt the features from previous work by Han *et al.* (2006), Tetreault and Chodorow (2008), and Rozovskaya *et al.* (2011) for our system. Tetreault *et al.* (2010) show that parse features can further increase performance, and we use the dependency parse features based on their work. For all the above features, the observed article or preposition used by the writer is "blanked out" when computing the features. However, we add the observed article or preposition as an additional feature for determiner and replacement preposition correction.

The features described so far are all binary-valued, i.e., they indicate whether some feature is present in the input or not. Additionally, we can construct real-valued features by counting the log frequency of surface N-grams on the web or in a web-scale corpus (Bergsma et al., 2009). Web-scale N-gram count features can harness the power of the web in connection with supervised classification and have successfully been used for a number of NLP generation and disambiguation problems (Bergsma et al., 2009; Bergsma et al., 2010), although we are not aware of any previous application in grammatical error correction. Web-scale N-gram count features usually use N-grams of consecutive tokens. The release of web-scale parsed corpora like the WaCky project (Baroni et al., 2009) makes it possible to extend the idea to dependency N-grams of child-parent tuples over the dependency arcs in the dependency parse tree, e.g., {(child, node), (node, parent)} for bigrams, {(child's child, child, node), (child, node, parent), (node, parent, parent's parent)} for trigrams. We collect log frequency counts for dependency N-grams from a large dependency-parsed web corpus and use the log frequency count as a feature. We normalize all real-valued feature values to a unit interval $[0, 1]$ to avoid features with larger values dominating features with smaller values.

4 Experiments

In this section, we report experimental results of our system on two different data sets: a held-out test split of the HOO 2012 training data, and the official HOO 2012 test set.

4.1 Data Sets

The HOO 2012 training data consists of 1,000 documents together with gold-standard annotation. The documents are a subset of the 1,244 documents in the Cambridge Learner Corpus FCE (First Certificate in English) data set (Yannakoudakis et al., 2011). The HOO 2012 gold-standard annotation only contains edits for six determiner and preposition error types and discards all other gold edits

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

Table 1: Features for determiner correction. Example: "The cat sat on *the black door mat*." † : lexical tokens in lower case, ‡: lexical tokens in both original and lower case

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

Table 1: (continued)

from the original FCE data set. This can lead to "wrong" gold edits that produce ungrammatical sentences, like the following sentence

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

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

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

Table 2: Features for replacement preposition correction. Example: "He saw a cat sitting *on the mat*." †: lexical tokens in lower case, ‡: lexical tokens in both original and lower case

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

Table 3: Features for missing preposition correction. Example: "He saw a cat sitting *the mat*."† : lexical tokens in lower case, ‡: lexical tokens in both original and lower case

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

Table 4: Features for unwanted preposition correction. Example: "The cat went *to home*."

Data set	# Documents	# Sentences	# Tokens
FCE(1044)	1,044	22,434	339,902
FCE(1244)	1,244	28,033	423,850
HOO-DEV	100	2,798	42,347
HOO-DEVTEST	100	2,674	41,518
HOO-TEST	100	1,393	20,563

Table 5: Overview of the data sets.

official HOO 2012 test data (HOO-TEST), which is not part of the FCE corpus, is completely unobserved during system development. Table 5 gives an overview of the data. Besides the FCE and HOO 2012 data sets, we use the following corpora. The Google Web 1T 5-gram corpus (Brants and Franz, 2006) is used for language modeling and collecting N-gram counts, the PukWaC corpus from the WaCky project (Baroni et al., 2009) is used for collecting web-scale dependency N-gram counts, and the New York Times section of the Gigaword corpus[3] is used for training the re-casing model. All data sets used in our system are publicly available.

4.2 Resources

We use the following NLP resources in our system. Sentence splitting is performed with the NLTK toolkit.[4] For spelling correction, we use the free software Aspell.[5] All words that appear at least ten times in the HOO 2012 training data are added to the spelling dictionary. We use the OpenNLP tools (version 1.5.2)[6] for POS tagging, YamCha (version 0.33) (Kudo and Matsumoto, 2003) for chunking, and the MaltParser (version 1.6.1) (Nivre et al., 2007) for dependency parsing. We use RandLM (Talbot and Osborne, 2007) for language modeling. The re-casing model is built with the Moses SMT system (Koehn et al., 2007) from the Gigaword New York Times section and all normal-cased documents in the HOO 2012 training data. The CuVPlus English dictionary (Mitton, 1992) is used to determine the countability of nouns. The CW learning algorithm is implemented by our group. The source code is available from our website.[7] All resources used in our system are publicly available.

4.3 Evaluation

Evaluation is performed by computing detection, recognition, and correction F_1 score between the set of system edits and the set of gold-standard edits as defined in the HOO 2012 overview paper (Dale et al., 2012). Detection scores are very similar to recognition scores (about 1–2% higher). We omit detection scores due to space limitations. Evaluation on the official test set is performed with respect to two different gold standards: the original gold standard from Cambridge University Press and a revised version which was created in the HOO 2012 shared task in response to change requests from participating teams. All scores are computed with the official scorer. The official gold-standard edits are given in character offsets, while our system internally works with token offsets. Therefore, all token offsets are automatically mapped back to character offsets before we submit our system edits. We only submitted one run of our system.

[3] LDC2009T13

[4] http://www.nltk.org

[5] http://aspell.net

[6] http://opennlp.apache.org

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

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

Table 6: Overall precision, recall, and F_1 score on the HOO-DEVTEST data after determiner correction (Det), replacement preposition correction (RT), and missing and unwanted preposition correction (MT/UT).

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

Table 7: Individual scores for each error type on the HOO-DEVTEST data.

4.4 Results

Tables 6 and 8 show the overall precision, recall and F_1 score of our system after each processing step on the held-out HOO-DEVTEST set and the official test set, respectively. All numbers are shown in percentages. We note that each processing step improves the overall performance. The final F_1 correction score on the official test set is 28.70% before revision and 37.83% after revision, which are the highest scores achieved by any participating team. Tables 7 and 9 show individual precision, recall, and F_1 score

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

(a) Before revisions

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

(b) After revisions

Table 8: Overall precision, recall, and F_1 score on the HOO-TEST data after determiner correction (Det), replacement preposition correction (RT), and missing and unwanted preposition correction (MT/UT).

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

(a) Before revisions

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

(b) After revisions

Table 9: Individual scores for each error type on the HOO-TEST data.

for each of the six error types, and for determiners (Det: aggregate of RD, MD, UD) and prepositions (Prep: aggregate of RT, MT, UT) on the held-out HOO-DEVTEST set and the official test set HOO-TEST, respectively.

5 Discussion

The main differences between our submission to the HOO 2011 shared task (Dahlmeier et al., 2011) and to this year's shared task are the use of the CW learning algorithm, the use of web-scale N-gram count features, and the use of the observed article or preposition as a feature. The CW learning algorithm performed slightly better than the empirical risk minimization batch learning algorithm that we have used previously while being significantly faster during training. Adding the web-scale N-gram count features showed significant improvements in initial experiments. Using the observed article or preposition feature allows the classifier to learn a bias against unnecessary corrections. We believe that our good precision scores are a result of using this feature.

In our experiments, we tried adding additional training data from other text corpora: the NUS Corpus of Learner English (NUCLE) (Dahlmeier and Ng, 2011) and the Gigaword corpus. Unfortunately, we did not see any consistent improvements over simply using the FCE corpus. The general rule of thumb that "more data is better data" did not seem to hold true in this case. After the evaluation had completed, we also tried training on additional training data and tested the resulting system on the official test set but did not see improvements either. We believe that no improvements were obtained due to the similarity between the training and test data, since all of them are student essays written in response to question prompts from the Cambridge FCE exam.

6 Conclusion

We have presented the system from the National University of Singapore that participated in the HOO 2012 shared task. Our system achieves the highest correction F_1 score on the official test set among all 14 participating teams, based on gold-standard edits both before and after revision.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

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

S. Bergsma, D. Lin, and R. Goebel. 2009. Web-scale N-gram models for lexical disambiguation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 1507–1512, Pasadena, California, USA.

S. Bergsma, E. Pitler, and D. Lin. 2010. Creating robust supervised classifiers via web-scale N-gram data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 865–874, Uppsala, Sweden.

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

K. Crammer, M. Dredze, and A. Kulesza. 2009. Multi-class confidence weighted algorithms. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 496–504, Singapore.

D. Dahlmeier and H.T. Ng. 2011. Grammatical error correction with alternating structure optimization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 915–923, Portland, Oregon, USA.

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

R. Dale, I. Anisimoff, and G. Narroway. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In *Proceedings of the Seventh Workshop on Innovative Use of NLP for Building Educational Applications*, Montréal, Québec, Canada.

M. Dredze, K. Crammer, and F. Pereira. 2008. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning*, pages 184–191, Helsinki, Finland.

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

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

T. Kudo and Y. Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 24–31, Sapporo, Japan.

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

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

A. Rozovskaya and D. Roth. 2010. Training paradigms for correcting errors in grammar and usage. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the ACL*, pages 154–162, Los Angeles, California.

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

D. Talbot and M. Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 512–519, Prague, Czech Republic.

J. Tetreault and M. Chodorow. 2008. The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 865–872, Manchester, UK.

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

H. Yannakoudakis, T. Briscoe, and B. Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA.