The AMU System in the CoNLL-2014 Shared Task: Grammatical Error Correction by Data-Intensive and Feature-Rich Statistical Machine Translation

The AMU System in the CoNLL-2014 Shared Task: Grammatical Error Correction by Data-Intensive and Feature-Rich Statistical Machine Translation

Marcin Junczys-Dowmunt **Roman Grundkiewicz**
Information Systems Laboratory
Adam Mickiewicz University
ul. Umultowska 87, 61-614 Poznań, Poland
{junczys,romang}@amu.edu.pl

Abstract

Statistical machine translation toolkits like Moses have not been designed with grammatical error correction in mind. In order to achieve competitive results in this area, it is not enough to simply add more data. Optimization procedures need to be customized, task-specific features should be introduced. Only then can the decoder take advantage of relevant data.

We demonstrate the validity of the above claims by combining web-scale language models and large-scale error-corrected texts with parameter tuning according to the task metric and correction-specific features. Our system achieves a result of 35.0% $F_{0.5}$ on the blind CoNLL-2014 test set, ranking on third place. A similar system, equipped with identical models but without tuned parameters and specialized features, stagnates at 25.4%.

1 Introduction

There has been an increasing interest in using statistical machine translation (SMT) for the task of grammatical error correction. Among the 16 teams that took part in the CoNLL-2013 Shared Task (Ng et al., 2013), four teams described approaches that fully or partially used SMT in their system. While in the previous year the correction task was restricted to just five error types, the CoNLL-2014 Shared Task (Ng et al., 2014) now requires a participating system to correct all 28 error types present in NUCLE (Dahlmeier et al., 2013). Since the high number of error types has made it harder to target each error category with dedicated components, SMT with its ability to learn generic text transformations is now an even more appealing approach.

With out-of-the-box machine translation toolkits like Moses (Koehn et al., 2007) being freely available, the application of SMT to grammatical error correction seems straightforward. However, Moses has not been designed as a grammar correction system, the standard features and optimization methods are geared towards translation performance measured by the metrics used in the SMT field. Training Moses on data that is relevant for grammatical error correction is a step in the right direction, but data alone is not enough. The decoder needs to be able to judge the data based on relevant features, parameter optimization needs to be performed according to relevant metrics.

This paper constitutes the description of the Adam Mickiewicz University (AMU) submission to the CoNLL-2014 Shared Task on Grammatical Error Correction. We explore the interaction of large-scale data, parameter optimization, and task-specific features in a Moses-based system. Related work is presented in the next section, the system setup is shortly described in Section 3. Sections 4 to 7 contain our main contributions.

In Section 4, we describe our implementation of feature weights tuning according to the MaxMatch (M^2) metric by Dahlmeier and Ng (2012b) which is the evaluation metric of the current CoNLL-2014 Shared Task. Sections 5 and 6 deal with the data-intensive aspects of our paper. We start by extending the baseline system with a Wikipedia-based language model and finish with a web-scale language model estimated from CommonCrawl data. Uncorrected/corrected data from the social language learner's platform Lang-8 is used to extend the translation models of our system.

Task-specific dense and sparse features are introduced in Section 7. These features are meant to raise the "awareness" of the decoder for grammatical error correction. In Section 8, we discuss the results of our submission and several intermediate systems on the blind CoNLL-2014 test set.

Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pages 25–33,
Baltimore, Maryland, 26-27 July 2014.

2 Related Work

Brockett et al. (2006) use SMT to correct countability errors for a set of 14 mass nouns that pose problems to Chinese ESL learners. For this very restricted task they achieve a results of 61.81% corrected mistakes. This work mentions minimum error rate tuning according to BLEU.

A Moses-based system is described by Mizumoto et al. (2011) who correct grammatical errors of learners of Japanese. This work is continued for English in Mizumoto et al. (2012). The effect of learner corpus size on various types of grammatical errors is investigated. The additional large-scale data originates from the social learner's platform Lang-8. We use similar resources.

Very interesting work is presented by Dahlmeier and Ng (2012a). A custom beam-search decoder for grammatical error correction is introduced that incorporates discriminative classifiers for specific error categories such as articles and prepositions. The authors perform parameter tuning and find PRO to work better with M_1^2 than MERT[1]. The specialized decoder tuned with M_1^2 is compared to Moses that has been tuned with BLEU. As we show in Section 4.2, this cannot be a fair comparison.

The CoNLL-2013 Shared Task (Ng et al., 2013) saw a number of systems based entirely or partially on translation approaches. Most notable are Yuan and Felice (2013) and Yoshimoto et al. (2013). Yuan and Felice (2013) apply Moses to all five error types of the shared task and extend the provided training data by adding other learner's corpora. They also experiment with generating artificial errors. Improvement over the baseline are small, but their approach to generate errors shows promise. We successfully re-implement their baseline. Yoshimoto et al. (2013) use Moses for two error classes, prepositions and determiners, for other classes they find classifier-based approaches and treelet language models to perform better. None of the CoNLL-2013 SMT-based systems seems to use parameter tuning.

3 General System Setup

Our system is based on the phrase-based part of the statistical machine translation system Moses (Koehn et al., 2007). Only plain text data is used for language model and translation model training. External linguistic knowledge is introduced during parameter tuning as the tuning metric relies on the error annotation present in NUCLE. Phrase tables are binarized with the compact phrase table (Junczys-Dowmunt, 2012), no reordering models are used, the distortion limit is set to 0, effectively prohibiting any reordering. Apart from that, our basic setup is very similar to that of Yuan and Felice (2013). We adapted their 4-fold cross validation scheme on NUCLE to our needs and use a similar baseline, now with 28 error types.

4 Parameter Tuning

The training of feature functions like translation models or language models is only half the work required to produce a state-of-the-art statistical machine translation system. The other half relies on parameter tuning.

During translation, Moses scores translations e of string f by a log-linear model

$$\log p(e|f) = \sum_i \lambda_i \log(h_i(e, f))$$

where h_i are feature functions and λ_i are feature weights. Without parameter tuning, results may be questionable as the choice of feature function weights (everything else being identical) can turn a mediocre system into a high-scoring system or render a good system useless. This is illustrated in Section 4.2 and by examples throughout the paper.

All our modifications to MERT, PRO, kb-MIRA discussed in this section are publicly available[2].

4.1 Tuning Scheme

To accommodate for parameter tuning, we modify the standard 4-fold cross validation procedure. The test set in each of the four training/testing runs is again divided into two halves. The first half is treated as a tuning set, the second half as a test set. Next, tuning set and test set are inverted in order to tune and test a second time. Altogether, we perform four separate translation model training steps and eight tuning/testing steps. Each tuning/test set consists of ca. 7,000 sentences. We call this procedure 4×2-fold cross validation (4×2-CV). This way the entire NUCLE corpus serves as training, test, and tuning set. We also evaluate all our results on the CoNLL-2013 gold standard (ST-2013) which has been made available with 28 error types after the previous shared task.

[1] This is different from our findings for Moses, but may be a property of their custom decoder.

[2] https://github.com/moses-smt/mosesdecoder/fscorer

	4×2-CV		ST-2013	
Tuned with	BLEU	$M^2_{0.5}$	BLEU	$M^2_{0.5}$
Untuned	85.52	14.02	70.38	19.05
BLEU	88.31	1.27	72.62	1.12
$M^2_{0.5}$	87.76	15.43	71.99	16.73
Original	89.51	0.00	72.67	0.00

Table 1: Tuning with BLEU and M^2

System	Concat.	Average
NUCLE	15.16	15.43
NUCLE+CCLM	22.03	22.19
Final	25.93	26.26

Table 2: Effects of parameter weight smoothing on three selected systems for 4×2-CV (CoNLL-2014)

4.2 Tuning Metric

We refer to $F_{0.5}$ computed by the M^2 metric as $M^2_{0.5}$. Moses is bundled with several tuning tools that can tune parameter vectors according to different MT tuning metrics. The most widely used is BLEU (Papineni et al., 2002). We first attempt minimum error rate tuning (MERT) (Och, 2003) with BLEU, results are shown in Table 1. While BLEU scores increase on both, 4×2-CV and ST-2013, the effect on $M^2_{0.5}$ is catastrophic[3] though not surprising. The baseline is so weak that it introduces more errors than corrections, thus lowering the similarity of the output and the reference below the level of the similarity of the input and the reference. MERT learns parameter weights that disable nearly all correction attempts.

The obvious solution is to tune directly with M^2. M^2 provides per-sentence sufficient statistics and can easily[4] be integrated with MERT. We retune with M^2 and see an improvement on 4×2-CV but a significant decrease for ST-2013. BLEU increases for this system despite the drop in M^2.

This seems contradictory, but actually proves our point about the necessity of parameter tuning. Good luck should not be a basis for choosing parameters, in the case of a blind submission we have a much better chance to reach good results betting on optimized parameters. As we see later, this situation does not occur again for the more advanced systems, tuned parameters do generally better.

4.3 Parameter Smoothing

Based on the results of Clark et al. (2011), it has become good practice to tune systems between three and five times and report average results in order to cope with optimizer instability. Cettolo et al. (2011) expand on this work and explore parameter smoothing methods for different parameter vectors obtained on the same tuning sets. They report that parameter vector centroids averaged over several tuning runs yield better than average results and reduce variation. Tuning three to five times would require 24 to 40 tuning runs in our setup. However, we already have eight parameter vectors obtained from distinct tuning sets and decide to average these parameters. This way we hope to obtain a single vector of smoothed parameters that represents the entire NUCLE corpus.

Eventually, we retranslate the test sets according to 4-fold cross validation using the respective training data with this parameter vector. The same parameters are later used with the full training data to translate the CoNLL-2013 test set and the blind CoNLL-2014 test set. As it turns out, averaging parameter vectors across all parts has a consistently positive effect for M^2. This is shown in Table 2, systems mentioned in the table are introduced in Section 5 and Section 7.2.

4.4 Tuning Sparse Feature Weights

Tuning sparse features (Section 7.2) with M^2 poses an unexpected challenge. Moses implements two methods for feature-rich tuning: PRO (Hopkins and May, 2011) and Batch k-best MIRA (kb-MIRA) (Cherry and Foster, 2012) that both function as drop-in replacements for MERT. MERT cannot be used directly with sparse features. When BLEU is used as a tuning metric, Koehn and Haddow (2012) report results for PRO on a par with MERT for a system with only dense features. Unfortunately, this cannot be confirmed for M^2; we consistently see worse results than for MERT using PRO or kb-MIRA.

PRO and kb-MIRA operate on sentence-level while MERT computes M^2 for the complete corpus. Similar to Dahlmeier and Ng (2012a), we use sentence-level M^2 as an approximation. We suspect that M^2 might not be distinctive enough in a

[3]Which might explain why none of the Moses-based CoNLL-2013 systems used parameter tuning.

[4]We run the original m2scorer Python code with an embedded Python interpreter in MERT's C++ source code.

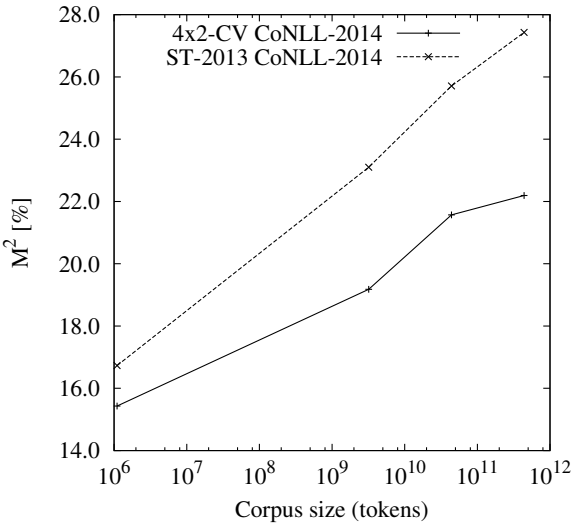

Figure 1: Language model corpus size versus M^2

sentence-based scenario.

Koehn and Haddow (2012) also explore a method they call "PRO-MERT" where PRO and MERT are run in turns. The parameter vector calculated by PRO serves as a starting point for MERT which optimizes dense features and in the case of existing sparse features a scalar weight that is multiplied with all sparse feature weights.

While this method does not seem to have any advantage for BLEU-based tuning in a MT setting it has a positive effect on tuning with M^2. Results for sparse features are now not worse than when tuned with MERT alone in a dense feature scenario. Additionally to "PRO-MERT", we implemented "kb-MIRA-MERT" which seems to display better convergence. As in the case of dense feature functions, we smooth sparse feature weights by averaging over all eight tuning steps.

All reported results in this paper have been tuned according to $M^2_{0.5}$, systems with dense features use MERT, systems with sparse features kb-MIRA-MERT. All results are given for parameter vectors that have been smoothed over eight optimizer runs from 4×2-CV.

5 Adding Language Model Data

With parameter tuning working, we can now explore the effects of adding feature functions to our system, starting with bigger language models.

All systems use one 5-gram language model that has been estimated from the target side of the parallel data available for training. In this section, only NUCLE is used as parallel data, four times 3/4 of NUCLE for 4×2-CV and complete NUCLE for ST-2013. If additional parallel data is added to the training process (see Section 6), the target data is concatenated with NUCLE and a new 5-gram language model is estimated.

System	4×2-CV	ST-2013
NUCLE	15.43	16.73
+WikiLM	19.18	23.10
+CCLM$_{10\%}$	21.57	25.71
+CCLM	22.19	27.43

Table 3: Results for increasing language models size on both shared task scenarios

The additional language models discussed in this section form separate feature functions, i.e. they are weighted separately from the target data language model. We experiment with three models that have been estimated using KenLM's (Heafield, 2011) modified Kneser-Ney estimation tool (Heafield et al., 2013):

WikiLM – a 3-gram model estimated from the entire English Wikipedia (2014-01-02). The raw text corpus consists of 3.2×10^9 tokens.

CCLM$_{10\%}$ – a 3-gram model estimated from 10% of the English CommonCrawl data (4.4×10^{10} tokens) described by Buck et al. (2014). The full corpus data has been made publicly available by the authors.

CCLM – a 5-gram model estimated from the entire CommonCrawl data (4.4×10^{11} tokens). This model has been created and made available to us by Kenneth Heafield. A newer version is publicly available (Buck et al., 2014).

Results are shown in Table 3. Improvements seem to be proportionate to the order of magnitude of the language model training corpora (Figure 1). $M^2_{0.5}$ improves by nearly 7% for 4×2-CV and by more than 10% on ST-2013.

6 Adding Translation Model Data

SMT systems for grammatical error correction can be trained on unannotated data. For the 28 error-type task from CoNLL-2014, we do not need the linguistically rich error annotations present in NUCLE to add more training data. It suffices to have parallel data in which the source text contains errors and the target text has been corrected. For English, such data is available.

System	4×2-CV	ST-2013
NUCLE+CCLM	22.19	27.43
+L8-NAIST	23.34	31.20
+L8	25.02	33.52
NUCLE+CCLM	17.50	29.01
+L8-NAIST	14.54	30.84
+L8	17.48	30.14

Table 4: Adding parallel data from Lang-8. Top results are for tuned systems, bottom results for untuned systems.

6.1 Lang-8

Mizumoto et al. (2011) published[5] a list of learner's corpora that were scraped from the social language learning site Lang-8 (`http://lang-8.com`). For our first experiments we use entries from "Lang-8 Learner Corpora v1.0" with English as the learned language, we do not care for the native language of the user. Only entries for which at least one sentence has been corrected are taken into account. Sentences without corrections from such entries are treated as error-free and mirrored on the target side of the corpus. Eventually, we obtain a corpus of 2,567,969 sentence pairs with 28,506,540 tokens on the uncorrected source side. No noise-filtering is applied. We call this resource L8-NAIST. Yoshimoto et al. (2013) use this resource for sub-elements of their system at the CoNLL-2013 Shared Task, but end up with half the number of sentences. This seems to be caused by noise-reduction.

We further investigate the effect of adding even greater parallel resources. Lang-8 is scraped for additional entries and we manage to nearly double the size of the corpus to 3,733,116 sentences with 51,259,679 tokens on the source side. This joint resource is labeled L8.

During training, the additional data is concatenated with all training corpora in our setup (3/4 of NUCLE for 4×2-CV and all of NUCLE for the final system).

Results are presented in Table 4. We extend the previous best system NUCLE+CCLM with L8-NAIST and L8. For tuned systems (top), results improve for both evaluation settings with growing corpus size. In the case of untuned systems (bottom) results are entirely inconclusive.

[5] `http://cl.naist.jp/nldata/lang-8`

6.2 Error Selection

Yuan and Felice (2013) generate artificial errors to add more training data to their system. We prefer actual errors, but the Lang-8 data may be too error-prone as the general level of proficiency seems to be lower than that of the NUCLE essays. We therefore select errors that match NUCLE error types and replace all other errors with their corresponding corrections.

For each pair of sentences, a sequence of deletions and insertions is computed with the LCS algorithm (Maier, 1978) that transform the source sentence into the target sentence. Adjacent deleted words are concatenated, adjacent inserted words result in a phrase insertion. A deleted phrase followed directly by a phrase insertion is interpreted as a phrase substitution. Substitutions are generalized if they consist of common substrings. Generalizations are encoded by the regular expression `(\w{3,})` and a back-reference, e.g. `\1`. Table 5 contains the 20 most frequent patterns extracted from NUCLE, 666 patterns with a frequency of five or higher remain. Next, we perform the same computation for the to-be-adapted data. Edits that match patterns from our list are kept, other edits are replaced with their corrections.

Although results (Table 6) with error selection increase for 4×2-CV, the NUCLE+CCLM+L8A seems to generalize poorly to new data, there is a significant drop for the external test set. Compared to NUCLE+CCLM+L8 (prec.: 59.80, rec.: 15.95) the error adapted (prec.: 70.07, rec.: 8.52) is much more conservative.

Inspired by this, we also try a combination (NUCLE+CCLM+L8AT as in Adapted Tuning) of both systems by tuning with the adapted NUCLE+CCLM+L8A, but applying the weights to the unadapted system NUCLE+CCLM+L8. This results in a gain of 5% for ST-2013. It seems that the unadapted Lang8 data introduces a substantial amount of noise that interferes with the tuning process. Weights obtained from the cleaned data seem to better approximate the true weight vector and also work with unadapted data without sacrificing recall. In the remainder of the paper we use this training/tuning scheme for all newly introduced systems.

7 Task-Specific Features

The systems presented so far relied on default features available in Moses. In this section we will

Pattern	Freq.	Pattern	Freq.
sub(«(\w{3,})»,«\1s»)	2441	ins(«an»)	222
ins(«the»)	2364	sub(«(\w{3,})d»,«\1»)	181
del(«the»)	1624	del(«of»)	178
sub(«(\w{3,})s»,«\1»)	1110	sub(«is»,«are»)	166
ins(«,»)	961	ins(«of»)	166
ins(«a»)	663	del(«a»)	160
sub(«(\w{3,})»,«\1d»)	253	sub(«(\w{3,})y»,«\1ies»)	150
del(«,»)	244	ins(«to»)	148
del(«.»)	227	sub(«is»,«was»)	147
sub(«(\w{3,})»,«\1ed»)	222	sub(«the»,«a»)	132

Table 5: 20 most frequent patterns extracted from NUCLE 3.0

System	4×2-CV	ST-2013
NUCLE+CCLM+L8	25.02	33.52
NUCLE+CCLM+L8A	26.82	28.67
NUCLE+CCLM+L8AT	26.82	38.59

Table 6: Results of error selection

Source (s)	Target (t)	$e^{d(s,t)}$
a short time .	short term only .	20.0855
a situation	into a situation	2.7183
a supermarket .	a supermarket .	1.0000
able	unable	2.7183

Table 7: Dense Levenshtein feature examples.

extend the translation model with features tailored to the task of grammatical error correction.

7.1 Dense Features

In Moses, translation models are described by a set of dense features: phrase translation probabilities, lexical scores, and a phrase penalty (Koehn et al., 2003). In the grammatical error correction scenario where source and target phrases are often identical or similar, it might be useful to inform the decoder about the differences in a phrase pair.

We extend translation models with a word-based Levenshtein distance feature (Levenshtein, 1966) that captures the number of edit operations required to turn the source phrase into the target phrase. Each phrase pair in the phrase table is scored with $e^{d(s,t)}$ where d is the word-based distance function, s is the source phrase, t is the target phrase. The exponential function is used because Moses relies on a log-linear model. In the log-linear model, the edit distances of all phrase pairs used to translate a sentence sum to the total number of edits that have been applied to produce the target sentence. Note that the Lang-8 data has not been processed for noise-reduction, this feature should take care of the problem and penalize sentences that have diverged to much from the source. Table 7 contains examples of phrase pairs and their Levenshtein distance feature.

We extend the currently best system NUCLE+CCLM+L8AT with the Levenshtein distance feature. Results are shown in Table 8 (+LD). For 4×2-CV small improvements can be observed, the effect is more significant for ST-2013. It can be concluded that this very simple modification of the standard translation model is a beneficial extension of SMT for grammatical correction.

7.2 Sparse Features

Sparse features are a relatively new addition to Moses (Hasler et al., 2012). Unlike dense features, they are optional and unrestricted in number, thousands of different sparse features may be used. A verbose version of the above mentioned LD feature is implemented as a sparse feature. Each edit operation is annotated with the operation type and the words that take part in the operation. The decoder can now learn to favor or penalize specific edits during tuning. As before in the case of error adaption patterns from Section 6.2, we generalize substitution operations if common substrings of a length equal to or greater than three characters appear in corresponding source and target phrases. In the end,

System	4×2-CV	ST-2013
NUCLE+CCLM+L8AT	26.82	38.59
+LD	27.34	40.21
+SF	**27.58**	**40.60**

Table 8: Results for dense Levenshtein distance (LD) and sparse pattern features (SF). Each component extends the previous system cumulatively.

we obtain sparse features that look exactly like these patterns. Features that correspond to patterns that had a frequency below 5 in NUCLE are mapped to `del(OTHER)`, `ins(OTHER)`, and `sub(OTHER1,OTHER2)`. Contrary to the Levenshtein distance feature, the sparse features are computed during decoding.

Sparse features are added to the system which has already been extended with the dense Levenshtein feature. Results in Table 8 (+SF) show small, but consistent gains. LD and SF are linearly dependent as the total sum of triggered sparse features should be equal to the value of LD for a sentence, but we still observe positive effects. Sparse feature tuning is currently a work-around with dubious effects, it can be expected that results might be more significant once this problem is solved. Based on these results, we choose the last system NUCLE+CCLM+L8AT+LD+SF as our final system for the CoNLL-2014 Shared Task.

8 Results for blind CoNLL-2014 test set

Our final system achieves an official result of 35.01% $M^2_{0,5}$ ("Submission" in Table 9) on the blind CoNLL-2014 Shared Task test set (ST-2014). Due to a tight time frame, this system suffered from missing words in an incorrectly filtered language model and too few tuning iterations. After the submission we retrained the same system and achieve a score of 35.38% $M^2_{0,5}$. Table 9 contains the results for incrementally added features, starting with the baseline, ending with the final system. The addition of a web-scale language model results in similar improvements as for 4×2-CV and ST-2013. Additional unadapted parallel training data from Lang-8 (+L8) has a very modest effect on ST-2014. This improves with the mixed tuning scheme (+L8AT) which shows that the gains for ST-2013 are not a one-time effect. Surprising are the substantial gains due to the dense Levenshtein feature and the sparse features. We suspect that the task-specific features allow the decoder to better exploit the potential of the Lang-8 data. This is verified by training NUCLE+CCLM+LD+SF which scores only 25.82%.

System	P	R	$M^2_{0.5}$
Submission	**41.62**	**21.40**	**35.01**
NUCLE	49.85	5.19	18.32
+CCLM	50.39	9.90	27.72
+L8	37.67	14.07	28.21
+L8AT	37.02	17.94	30.53
+LD	39.41	22.15	34.10
+SF	**41.72**	**22.00**	**35.38**
NUCLE	36.59	9.96	23.84
+CCLM	27.92	18.68	25.41
+L8	25.06	26.75	25.38
+L8AT	24.49	34.89	26.04
+LD	25.94	36.41	27.52
+SF	25.94	36.41	27.52

Table 9: Performance of chosen systems on the CoNLL-2014 test set. Bottom results are untuned.

To support our claim concerning the importance of parameter tuning, we also provide the performance of the same systems on ST-2014 with standard parameters (bottom of Table 9). With one exception, we see significant improvements with tuning. The untuned systems display very similar results which would make it difficult to choose among the configurations (untuned +LD and +LD+SF are actually the same system). One might conclude incorrectly that the new features and additional resources have very little effect on the final results and miss a gain of ca. 8%.

Table 10 contains the ranking for all participating systems. Our system ranks on third place (see the Shared Task proceedings (Ng et al., 2014) for more information on the other systems), loosing by 2.32% and 1.78% against the first two teams. We win with a quite significant margin of 4.13% over the next best system. Compared to the top-two systems we suffer from lower recall, a problem which should be attacked in the future.

Participants were invited to submit alternative answers for evaluation, i.e. answers that were generated by their system and considered to be correct alternatives to the provided gold standard. These answers were checked by human annotators. Only three teams submitted alternative an-

Rank	Team ID	P	R	$M^2_{0.5}$
1	CAMB	39.71	30.10	37.33
2	CUUI	41.78	24.88	36.79
3	**AMU**	**41.62**	**21.40**	**35.01**
4	POST	34.51	21.73	30.88
5	NTHU	35.08	18.85	29.92
6	RAC	33.14	14.99	26.68
7	UMC	31.27	14.46	25.37
8	PKU	32.21	13.65	25.32
9	NARA	21.57	29.38	22.78
10	SJTU	30.11	5.10	15.19
11	UFC	70.00	1.72	7.84
12	IPN	11.28	2.85	7.09
13	IITB	30.77	1.39	5.90

Table 10: Shared Task results for submission without alternative answers. **AMU** is our result.

swers: CAMB, CUUI, and UMC. The results for all teams improved when evaluated on these additional answers, naturally those teams that submitted answers had the greatest gains. Our result with additional answers is 38.58%, we remain on third place after CUUI (45.57%) and CAMB (43.55%) which switched places. However, we do not consider the evaluation on alternative answers to be meaningful as it is strongly biased.[6]

9 Conclusions

We have shown that pure-surface phrase-based SMT can be used to achieve state-of-the-art results for grammatical error correction if sufficiently large resources are combined with correctly executed parameter tuning and task-specific features. For noisy data, it seems beneficial to tune on cleaned data, but noise can be useful when correcting unseen texts.

Most of the previous work that we reviewed lacked the detail of parameter tuning that is commonly applied in SMT. In consequence, potentially useful contributions rarely improved over the baselines or were beaten by classifier-based approaches. Many good features might have been overlooked or dismissed as unhelpful. Our findings invite to re-evaluate these previous results. The tools we extended for parameter tuning according to the M^2 metric are publicly available and we strongly suggest to use them in the future or to adapt them to the particular task at hand. Parameter tuning of sparse features according to the M^2 metric is ongoing research, but it seems the proposed work-around is a viable option.

Since it is quite simple to implement the task-specific features introduced in this paper, we recommend to use them whenever Moses is applied in a similar setting.

[6] We would accept alternative answers if all original system submissions were to be analyzed by annotators not associated with any team. If this is not possible due to considerable costs and efforts, we would advocate to abandon the current practice altogether.

References

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 249–256, Stroudsburg, USA. Association for Computational Linguistics.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the Common Crawl. In *Proceedings of the Language Resources and Evaluation Conference*, pages 3579–3584, Reykjavík, Iceland.

Mauro Cettolo, Nicola Bertoldi, and Marcello Federico. 2011. Methods for smoothing the optimizer instability in SMT. In *MT Summit XIII: the Thirteenth Machine Translation Summit*, pages 32–39.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Stroudsburg, USA. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, HLT '11, pages 176–181, Stroudsburg, USA. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012a. A beam-search decoder for grammatical error correction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 568–578, Stroudsburg, USA. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2012b. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies, pages 568–572, Stroudsburg, USA. 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*, pages 22–31, Atlanta, Georgia. Association for Computational Linguistics.

Eva Hasler, Barry Haddow, and Philipp Koehn. 2012. Sparse lexicalised features and topic adaptation for SMT. In *Proceedings of the 7th International Workshop on Spoken Language Translation (IWSLT)*, pages 268–275.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 187–197, Stroudsburg, USA. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1352–1362, Stroudsburg, USA. Association for Computational Linguistics.

Marcin Junczys-Dowmunt. 2012. Phrasal rank-encoding: Exploiting phrase redundancy and translational relations for phrase table compression. *Prague Bull. Math. Linguistics*, 98:63–74.

Philipp Koehn and Barry Haddow. 2012. Towards effective use of training data in statistical machine translation. In *Proceedings of the 7th Workshop on Statistical Machine Translation*, WMT '12, pages 317–321, Stroudsburg, USA. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54, Stroudsburg, USA. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710.

David Maier. 1978. The complexity of some problems on subsequences and supersequences. *Journal of the ACM*, 25(2):322–336.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated japanese error correction of second language learners. In *The 5th International Joint Conference on Natural Language Processing*, pages 147–155.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yu Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012*, pages 863–872.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the 17th Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

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)*, pages 1–14, Baltimore, USA. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, USA. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Stroudsburg, USA. Association for Computational Linguistics.

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 17th Conference on Computational Natural Language Learning: Shared Task*, pages 26–33, Sofia, Bulgaria. Association for Computational Linguistics.

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the 17th Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.