Statistical Machine Translation for Grammatical Error Correction

Arvind Srinivasan
Columbia University
vs2371@columbia.edu

Louis Cialdella
Columbia University
lmc2179@columbia.edu

Abstract

This paper details our experiments in using Statistical Machine translation methods for error detection in English as part of the Conll-13 (Johnson et. al, 2007) challenge. The challenge involved correction of errors made by Singaporean students of English for a variety of error types. The paper talks about our attempts to use Significance testing, Downsampling, and stemming in improving the output quality of the SMT system.

1 Task + Corpus

This project was initiated as a submission to the ConLL 2013 Shared Task: Grammatical Error Correction [1]. As a result, all training and test data was from a provided corpus, the National University Singapore Corpus of Learner English, or NUCLE, and trained systems were scored with the provided NUS MaxMatch, or M^2 scorer.

The corpus itself is comprised of 1400 annotated essays written by english-second-language Singaporean students. The annotations provide an error type and correction, from the below types of errors:

tag	category
Vt	Verb tense
Vm	Verb modal
V0	Missing verb
Vform	Verb form
SVA	Subject-verb-agreement
ArtOrDet	Article or Determiner
Nn	Noun number
Npos	Noun possesive
Pform	Pronoun form
Pref	Pronoun reference
Prep	Preposition
Wci	Wrong collocation/idiom
Wa	Acronyms
Wform	Word form
Wtone	Tone
Srun	Runons, comma splice
Smod	Dangling modifier
Spar	Parallelism
Sfrag	Fragment
Ssub	Subordinate clause
WOinc	Incorrect sentence form
WOadv	Adverb/adjective position
Trans	Link word/phrases
Mec	Punctuation, capitalization, spelling, typos
Rloc	Local redundancy
Cit	Citation
Others	Other errors
Um	Unclear meaning (cannot be corrected)

For the initial approach to this task, however, we looked to train a system that would identify a set of five errors: SVA, ArtOrDet, Nn, VForm, and Prep. The total corpus amounted to 67372 sentences, of which 15,000 were noisy (references, urls, etc).

[1] http://www.comp.nus.edu.sg/ nlp/conll13st.html

Of that, 11288 sentences had annotations with those tags, leaving a fairly small dataset on which to train. After the time of this writing, ConLL has released an additional dataset - the blind test data with the gold references, so we anticipate that this dataset will grow significantly.

2 Problem Background: SMT motivation

We begin by motivating the approach we take by considering the problem in a little more depth. Two of the most common types of error are article errors and preposition errors. Article errors are most common for speakers of languages which themselves lack any sort of article (some very common examples are Chinese, Japanese, and Russian), though they are naturally present in the writings of most all students. Similarly, preposition errors account for a large number of errors, due to the vast complexity of the English preposition system. Preposition use is highly particular, and is often governed by the context in which the preposition appears (making it a good candidate for phrase based correction).

An additional major source of error is that of collocation (or idiomatic) errors. These arise when there is a strong association between words even though other choices might be semantically or syntactically correct (for example, strong computer makes sense, but powerful computer is idiomatically rendered). Additionally, this type of error encompasses stock idiomatic phrases where the individual words are not obviously related to the actual meaning (to hit the nail on the head, to kick the bucket, etc). These phrases are usually non-modifiable in many contexts or cannot be modified and retain their meaning unless great care is used. While rule-based approaches lead to extremely complex definitions, a statistical approach is well suited to collocation errors since it provides an easy and concrete way of taking into account the relationships between words and/or whole idiomatic phrases.

3 Baseline

Our baseline system used Moses to ”translate” between a test corpus of original essays and the counterpart essays with all the annotations applied. This ”flattened” corpus was split into 58000 training sentences, 2000 sentences for tuning, and 8000 test sentences.

Moses used Giza++ to align the sentences, with default heuristics and reordering models (grow-diag-final, msd-bidirectional-fe). The tuning used MERT with default features.

To score the baseline, we use BLEU as well as the M^2 (Dahlmeier and Ng, 2012) scorer. The M^2 scorer scores precision, accuracy, and F1 against the annotations, rather than the text itself. Though the conference version of M^2 is case-sensitive, we use a case-insensitive version for simplicity - re-casing in English is a trivial task. Though both of these scorers support multi-reference evaluation, the corpus itself only has a single-reference gold reference file. Clearly, there are multiple equally fluent machine-generated correction candidates, even within the phrase table generated by the small training corpus, so we think this is an area that needs to be explored in terms of test corpus augmentation.

In reality, the optimality of a ”correction” would be gauged by fluency and grammatical cohesion of the final generation, so neither Bleu nor M^2 adequately capture the ideal result - an ideal result would likely be modeled by a combination of the two, with multiple references.

4 Issues with Baseline

The baseline displayed atrocious performance across the board. Two major and easily observable issues arise: the number of useless phrases in the phrase table, and the bias of phrases towards not making changes.

Firstly, we note that the vast majority of phrases in the phrase table are entirely unhelpful in decoding. For example, the number of possible translations of ”(” is obscenely large, having hundreds of entries. Of these, only a handful are useful, and even then the most common substitution should still be ”(”, as no errors using the left paren character occur anyway. This happens for virtually every piece of punctuation and trivial phrase, and does nothing except slow down decoding and cause spurious translation issues during decoding. We deal with a solution to this in the phrase table pruning section.

A second issue is that the phrase table entries are biased towards not making changes when decoding happens. That is, for a given phrase, the phrase table causes there to be a much larger chance of said phrase remaining unchanged in the final product, rather than changing and potentially correcting an error. A fairly typical PT entry indicative of this issue is:

, according to ——— , according to the ——— 0.166667 0.99065 0.0384615 0.578006 2.718 ——— 0-0 1-1 2-2 ——— 6 26 1

, according to ——— , according to ——— 0.961538 0.99065 0.961538 0.986281 2.718 ——— 0-0 1-1 2-2 ——— 26 26 25

We note here that the bottom phrase (which makes no changes) has a probability of 0.961538, while the top phrase has a probability of 0.0384615. We talk about our approach to this in the section on downsampling.

Additionally, we run into the problem of the BLEU metric, which causes problems with both tuning and evaluation. In this task, the BLEU metric used by Moses is quite a poor indicator of success. BLEU only checks for particular N-Grams in the reference, and in this task there is only a single reference sentence. This means that BLEU measures only N-grammatic distance to the supplied reference, which makes no guarantee of fluency. That is, a given change might drastically improve a sentence and make an otherwise incorrect sentence mostly or completely correct, but this may not be reflected in the BLEU score if it does not completely line up with the reference sentence. This leads to both strange tuning artifacts (such as the over-tuning described above) and difficult-to-interpret evaluation results.

5 Corpus cleaning and Datasets

The first immediate issue with the baseline phrase table was that references and urls generated a large amount of noise in the correction data. Since we used the wordpunct NLTK tokenization scheme, urls in particular were inconsistently treated as multiple words, creating noisy alignments as well. Thus, the first cleaning step was to eliminate references completely, re-adding them after the correction. We accomplished this with a simple reegex substitution.

Additionally, to experiment on the domain sensitivity of our system, we split the corpus by essay topic. This was particularly necessary because one of the challenges of the shared task was to submit system output for topics both inside and outside the training data. However, since the topics were not available a priori, we used an online version of the Latent Dirichlet Allocation (LDA) algorithm to generate a topic model, and then a K-Means clustering implementation to split the documents by those topics. We were then able to generate a "held out" dataset with 21572 training sentences (no references) with all but one topic, and 8695 test sentences (no references) with the remaining topic. This experiment showed that the NUCLE essay topics were largely similar in content, and that domain sensitivity is still a concern that ought to be tested by a held-out test set with a topic further removed from the training corpus.

To experiment with the impact of zero-annotation sentences in the training corpus, we also generated datasets without correct sentences. This left a dataset of 11288 sentences, of which we used 10000 to train and 288 to tune.

To run stemming experiments, we used the stemmers bundled with NLTK (Bird, 2007). We stemmed and lemmatized the two corpora mentioned above with the Lancaster and Snowball stemmers and the WordNet lemmatizer to experiment with various degrees of stemming aggressiveness.

We used a Moses (SRILM) ngram language model trained on the test and dev datasets. Given the topic closeness between the test and train sets and the relatively small size of the test set, there was not a significant OOV problem, and experiments with big language models trained on the Brown Corpus did not yield any benefits to either of the scoring metrics.

6 Approach

6.1 Significance Testing

We noted above that the Baseline model's phrase table contained a large number of ”junk” entries. One of our first challenges was eliminating such entries. A naive way of doing this is to choose a parameter; then, for all phrase tables with probability lower than this parameter, drop the phrase and redistribute its weight. For a number of reasons, this is an unwise and unstable way of doing this. Rather, we chose to use the method proposed by Johnson et. al (Johnson 2007), which uses tests of statistical significance to prune the phrase table. Ideally, this would remove statistically significant entries (such as the aforementioned left-paren nuisances).

6.2 Downsampling

We also noted above that there is a significan skew towards not changing particular phrases in the table. One way of dealing with this is by preprocessing out the sentences which are already correct and don't change, as mentioned in this paper. An additional way of dealing with this is by readjusting the distribution of phrases in the phrase table. Here, we choose to do this via a variable input parameter (which we'll denote α, where $\alpha \in [0, 1]$). For each phrase in a distribution over possible word translations, we multiply the probability of an entry that does not change by α and redistribute the missing weight over entries that do change. In this manner, we attempt to ameliorate the problems mentioned in the baseline section.

6.3 Stemming

One of the issues with longer phrase choice, especially with verb and SVA errors, is that of sparsit. In general, these are context determined errors, so the ideal scenario would have a phrase table with a single source phrase translating to a target phrase, rather than multiple candidates that represent the different error type.

Take, for example, the phrase ”The dog had walked.” With a verb or SVA error, this can be formulated as ”The dog had walks”, ”The dog had walk”, ”The dog had walking,” etc. However, there is only one valid annotation. In this case, it makes sense to reduce the word to its base form on the source side, because there will then be 4 training examples rather than one per type. In this trivial example, this also intuitively makes decoding less ambiguous, as reduction to base forms

Though this could be achieved with segmentation as well (and that could be a different experiment), we decided that stemming was more likely to yield a consistent representation of this theory. We experimented with various levels of stemming aggressiveness (the WordNet lemmatizer on the low end, and the Lancaster stemmer on the high end), using our different training sets to measure the impact.

6.4 NoCorrect - Only Errors

The primary issue with the baseline system was the bias towards a correct → correct translation in decoding. We tried to address this in various ways, but the naive way to do this was to train a system that dropped all sentences without annotations altogether, essentially forcing the SMT system to learn the annotation types with high probability. A variant of this system would experiment with various ”factors” of correctness in the training data, where each

We ran two experiments on this. The first was to test the performance on this system on a test set that only had sentences with annotations. The second was to test it on the full test set that we used for the other experiments. The former was to establish the hypothesis that decoding performance was dependent on a linear relationship between the number of ”errors” in the training sets and the test sets, and the latter was to stay consistent with the methodology of the other experiments. In practice, this method is infeasible - without another model to recognize correctness, there is no way to decode ”only sentences with errors” in a blind test, or production setting.

7 Results

All the experiments we ran achieved consistently high Bleu scores, but this includes the sentences that were already correct and mapped to another correct sentence. When the test set was scored with BLEU without any modifications, it achieved a BLEU score of of 87.32, 95.8/90.6/85.5/80.6.

Experiment	**Bleu**
Baseline	*96.34, 98.7/97.1/95.6/94.1*
Lancaster	*73.04, 88.2/77.6/68.6/60.6*
WordNet	*86.48, 94.1/88.8/84.1/79.6*
HeldOut	*96.81, 98.9/97.5/96.2/94.9*
NoCorrect	*96.46, 98.7/97.2/95.7/94.3*
Stem+NoCorr	*88.26, 95.8/90.8/86.0/81.6*
HeldOut+Sig	*96.80, 98.9/97.5/96.1/94.8*
NoCorr+Sig	*95.59, 98.5/96.5/94.6/92.8*
Lanc+Sig	*89.01, 96.3/91.2/87.0/82.9*
Word+Sig	*92.87, 97.4/94.4/91.6/88.8*

With the M^2 scorer, the different experiments yielded marginal improvements. In this case, the starting point for the score was 0.00, since the precision/recall/f1 were evaluated against only the generated and gold annotations.

M^2	*Precision*	Recall	F1
Baseline	*0*	0	0
Lancaster	*0.0215*	0.1915	0.0387
WordNet	*0.049*	0.1773	0.0768
HeldOut	*0.0116*	0.0663	0.0198
NoCorrect	*0.0838*	0.2555	0.1262
Stem+NoCorr	*0.0624*	0.3086	0.1038
HeldOut+Sig	*0.0086*	0.0312	0.0134
NoCorr+Sig	*0.0268*	0.1031	0.0425
Lanc+Sig	*0.0073*	0.0312	0.0118
Word+Sig	*0.0221*	0.0906	0.0355

For the significance testing experiments, a significant amount of the phrase table was pruned of statistically insignificant entries.

M2	*% of PT pruned*
HeldOut+Sig	*56.90%*
NoCorr+Sig	*53.17%*
Lanc+Sig	*51.81%*
Word+Sig	*57.57%*

8 Observations

8.1 Significance Testing

Firstly, we notice that a huge amount of the PT has been pruned, more than 50% in most cases. Second, we notice an almost monotone increase in BLEU score - which, while hard to interpret, does mean that significance testing moves the output closer to the reference. Sadly, we still see no very valueable improvement in the M2 score.

8.2 Downsampling

Downsampling was attempted at values of $\alpha = 0.7, \alpha = 0.9$, as those were the best of the handful of values we tried. The effect was hardly pronounced, and seems to make little difference in BLEU or M2 score. However, more thorough testing might have allowed us to find a more interesting pattern, and is a suitable goal for future work.

In particular, it seems that future work could be done by optimizing the α parameter, or by transforming the distribution in more interesting ways.

8.3 Stemming

The results from the stemming experiments were a mixed bag, and probably suffered from the fact that it was a broad-stroke approach when a finer brush might have been preferable. Take, for example, the following phrase table entries for the Lancaster experiment:

```
wait ||| to wait ||| 0.333333 1 0.047619 0.00710397 2.718 ||| 0-1 ||| 3 21 1
wait ||| wait ||| 1 1 0.428571 0.45 2.718 ||| 0-0 ||| 9 21 9
wait ||| waiting ||| 1 1 0.52381 0.55 2.718 ||| 0-0 ||| 11 21 11
```

```
’ s fin ||| ’s final ||| 1 0.25 0.0769231 0.15583 2.718 ||| 0-0 1-0 2-1 ||| 1 13 1
’ s fin ||| ’s finances ||| 1 0.25 0.384615 0.0249327 2.718 ||| 0-0 1-0 2-1 ||| 5 13 5
’ s fin ||| ’s financial ||| 1 0.25 0.538462
```

These two phrase table entries are indicative of a few characteristic trends of the stemming experiments:

1. The stemming had the intended effect on the verb entries in the phrase table, appropriately

recognizing different verb forms with roughly equal probability. The first example shows the rather interesting characteristic that the verb ”wait” is rarely corrected to the infinitive form, which, without stemming, would not have been found, as there would have been a single entry with ”to wait” → ”to wait” unless there was a specific annotation that added the infinitive.

2. The stemming had the opposite effect on nouns and adjectives, introducing ambiguity where there was previously none. In the second example, stemming to ”fin” meant that there were three candidates for a single base form, where only one was really valid in the context of another sentence.

3. An additional insight was that tokenized punctuation can either be helpful or specifically unhelpful in the context of error correction. In the second example, the {’ s} serves as potentially useful disambiguation - the following word can only be a noun, adjective, or adverb, since it is a possessive particle. However, parentheses, ellipses, and commas are unhelpful phrase boundaries, and should probably have been dropped altogether. Though we did not consider the punctuation error type for the initial tests, we anticipate that this would have been problematic had we done so.

Some of the additional noise introduced with stemming was fixed by being less aggressive (specifically the noise characteristic of the latter example), and in our tests, the WordNet lemmatizer performed best. In future tests (see sec. 10), we want to isolate error types in decoding sets to solidify the claim that stemming specifically increases recall and precision on Verb form and SVA errors, and does not help in the other cases - the alternative is that stemming merely introduces ambiguity in the same way downsampling does, and so it achieved slight gains just due to overcorrection.

8.4 NoCorrect - Only Errors

Though not included in the above table, the test of our NoCorrect system only decoding sentences with errors had the most encouraging results, as shown below:

BLEU = 88.26, 95.8/90.8/86.0/81.6
M^2 Precision: 0.4697
M^2 Recall : 0.1951
M^2 F1 : 0.2757

What this tells us is that at least for these 5 error forms, training a system on just errors is the best approach - it intuitively overcorrects in the same way that stemming or downsampling does artificially. The challenge then becomes twofold:

1. Increase the size of the training set significantly - after dropping incorrect sentences and noise, the training set was only 10000 sentences.

2. Drop sentences that are probably correct before decoding – as the results show, the performance significantly drops with the full training set.

Additionally, it remains to be seen how this approach generalizes to the other error types. The five error types that we dealt with are largely single word corrections, so the SMT system was at least able to learn the most common word substitutions. For longer error types, like fragments, runons, redundancy, and incorrect sentence form, the NoCorrect system would probably have much more damaging overcorrection issues.

9 Conclusions

Our results did not do much to validate SMT as a robust approach for error correction, as our experiments more or less failed across the board. Although these were only preliminary experiments, there were a few error types that seemed to have systemic issues when addressed in an SMT framework, and a few of the experiments (NoCorrect and Significance Testing in particular) were entirely designed to mitigate some of those systemic issues rather than introduce any particular linguistic nuance to the correction task. That said, part of this was self inflicted - the errors that we focused on for the shared task were those that may have been best served by a parse-tree rule based approach. The SMT approach, on the other hand, would have been able to find results for error types that we did not address due to sparsity of annotation, but are not tractable in a rule framework, like fragment correction or idiomatic consistency. We also avoided some tricky reordering issues that may have resulted from error types like redundancy, or anything that depended on related clauses.

That said, there is a significant body of SMT research that introduces syntactic information into the translation task, and SMT is clearly an important approach to pursue for error correction, even only because rules are highly language dependent, suffer significantly in the presence of noise, and do not account for the inherent ambiguity in the error correction task. We hope that our research guides other researchers to more interesting conclusions in the future.

10 Future Work

The results of this project were disappointing but not disheartening, and rather than concrete conlusions, we drew several insights that may lead to further research. These approaches fall into a few categories.

10.1 Pre-Editing

The NoCorrect model achieved fairly good results when decoding sentences with only errors, so one possible approach to error correction would be to train a model on incorrect sentences, and then use syntactic information to decide whether a sentence ”probably” needs a correction. This is analogous to ”pre-editing” with rules, but to avoid the need for written rules, which would obviate the need for SMT, that information could be incorporated in a large language model (Tan et. al, 2012) - high perplexity sentences would then be considered translation candidates.

10.2 Corpus Augmentation

One of the issues with even running meaningful experiments on the corpus was the lack of training data, and the small number of training examples of errors other than the 5 main types. However, none of the methods we used actually used the error types, and instead translated between flattened versions of the annotated corpora. This opens the door to augmenting the corpus - in tests, the closest documents stylistically to the NUCLE corpus were wikipedia articles in terms of level of grammar formality. One can use the topics generated from LDA topic analysis to generate a Wikipedia corpus, and then align it with a parallel foreign language corpus (Adafre et. al, 2006). The foreign language corpus can then be translated to come up with a aligned, incorrect → correct parallel corpus. The additional advantage of this approach is that it may detect idiosyncratic language errors with careful selection of the parallel Wikipedia language.

10.3 Tuning Modifications

Much of the motivation for statistical significance pruning and downsampling was that the tuning process overweighted BLEU correctness, which meant that correct → correct translations were prioritized. This could partially be solved by minimizing losses against the M^2 metric in combination with BLEU, either as features or otherwise.

10.4 Iterative Decoding

One of the consistent themes in the project was that different error types benefit from different approaches. This leads to the conclusion that it may be beneficial to train multiple systems on different error types, and then iteratively decode in a logical chain to get better performance.

Acknowledgments

We especially want to thank Professors Habash and Tomeh for their guidance and patience during this project. We both learned a lot, and are excited to continue investigating approaches to SMT and error correction in the future.

References

H. Johnson, J. Martin, G. Foster and R. Kuhn. 2007. *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pp. 967-975.*

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.*

Bird, Steven. 2007. *NLTK: the natural language toolkit. Proceedings of the COLING/ACL on Interactive presentation sessions.*

Adafre, Sisay Fissaha and De Rijke, Maarten. 2006. *Finding similar sentences across multiple languages* *in wikipedia. Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, pp. 62-69.*

Tan, Ming and Zhou, Wenli and Zheng, Lei and Wang, Shaojun. 2012. *A scalable distributed syntactic, semantic, and lexical language model. Computational Linguistics v38, 3, pp 631-671. MIT Press.*