Automated Whole Sentence Grammar Correction Using a Noisy Channel Model

Y. Albert Park
Department of Computer Science and Engineering
9500 Gilman Drive
La Jolla, CA 92037-404, USA
yapark@ucsd.edu

Roger Levy
Department of Linguistics
9500 Gilman Drive
La Jolla, CA 92037-108, USA
rlevy@ucsd.edu

Abstract

Automated grammar correction techniques have seen improvement over the years, but there is still much room for increased performance. Current correction techniques mainly focus on identifying and correcting a specific type of error, such as verb form misuse or preposition misuse, which restricts the corrections to a limited scope. We introduce a novel technique, based on a noisy channel model, which can utilize the whole sentence context to determine proper corrections. We show how to use the EM algorithm to learn the parameters of the noise model, using only a data set of erroneous sentences, given the proper language model. This frees us from the burden of acquiring a large corpora of corrected sentences. We also present a cheap and efficient way to provide automated evaluation results for grammar corrections by using BLEU and METEOR, in contrast to the commonly used manual evaluations.

1 Introduction

The process of editing written text is performed by humans on a daily basis. Humans work by first identifying the writer's intent, and then transforming the text so that it is coherent and error free. They can read text with several spelling errors and grammatical errors and still easily identify what the author originally meant to write. Unfortunately, current computer systems are still far from such capabilities when it comes to the task of recognizing incorrect text input. Various approaches have been taken, but to date it seems that even many spell checkers such as Aspell do not take context into consideration, which prevents them from finding misspellings which have the same form as valid words. Also, current grammar correction systems are mostly rule-based, searching the text for defined types of rule violations in the English grammar. While this approach has had some success in finding various grammatical errors, it is confined to specifically defined errors.

In this paper, we approach this problem by modeling various types of human errors using a noisy channel model (Shannon, 1948). Correct sentences are produced by a predefined generative probabilistic model, and lesioned by the noise model. We learn the noise model parameters using an expectation-maximization (EM) approach (Dempster et al., 1977; Wu, 1983). Our model allows us to deduce the original intended sentence by looking for the the highest probability parses over the entire sentence, which leads to automated whole sentence spelling and grammar correction based on contextual information.

In Section 2, we discuss previous work, followed by an explanation of our model and its implementation in Sections 3 and 4. In Section 5 we present a novel technique for evaluating the task of automated grammar and spelling correction, along with the data set we collected for our experiments. Our experiment results and discussion are in Section 6. Section 7 concludes this paper.

2 Background

Much of the previous work in the domain of automated grammar correction has focused on identi-

fying grammatical errors. Chodorow and Leacock (2000) used an unsupervised approach to identifying grammatical errors by looking for contextual cues in a ± 2 word window around a target word. To identify errors, they searched for cues which did not appear in the correct usage of words. Eeg-olofsson and Knutsson (2003) used rule-based methods to approach the problem of discovering preposition and determiner errors of L2 writers, and various classifier-based methods using Maximum Entropy models have also been proposed (Izumi et al., 2003; Tetreault and Chodorow, 2008; De Felice and Pulman, 2008). Some classifier-based methods can be used not only to identify errors, but also to determine suggestions for corrections by using the scores or probabilities from the classifiers for other possible words. While this is a plausible approach for grammar correction, there is one fundamental difference between this approach and the way humans edit. The output scores of classifiers do not take into account the observed erroneous word, changing the task of editing into a fill-in-the-blank selection task. In contrast, editing makes use of the writer's erroneous word which often encompasses information neccessary to correctly deduce the writer's intent.

Generation-based approaches to grammar correction have also been taken, such as Lee and Seneff (2006), where sentences are paraphrased into an over-generated word lattice, and then parsed to select the best rephrasing. As with the previously mentioned approaches, these approaches often have the disadvantage of ignoring the writer's selected word when used for error correction instead of just error detection.

Other work which relates to automated grammar correction has been done in the field of machine translation. Machine translation systems often generate output which is grammatically incorrect, and automated post-editing systems have been created to address this problem. For instance, when translating Japanese to English, the output sentence needs to be edited to include the correct articles, since the Japanese language does not contain articles. Knight and Chander (1994) address the problem of selecting the correct article for MT systems. These types of systems could also be used to facilitate grammar correction.

While grammar correction can be used on the output of MT systems, note that the task of grammar correction itself can also be thought of as a machine translation task, where we are trying to 'translate' a sentence from an 'incorrect grammar' language to a 'correct grammar' language. Under this idea, the use of statistical machine translation techniques to correct grammatical errors has also been explored. Brockett et al. (2006) uses phrasal SMT techniques to identify and correct mass noun errors of ESL students. Désilets and Hermet (2009) use a round-trip translation from L2 to L1 and back to L2 to correct errors using an SMT system, focusing on errors which link back to the writer's native language.

Despite the underlying commonality between the tasks of machine translation and grammar correction, there is a practical difference in that the field of grammar correction suffers from a lack of good quality parallel corpora. While machine translation has taken advantage of the plethora of translated documents and books, from which various corpora have been built, the field of grammar correction does not have this luxury. Annotated corpora of grammatical errors do exist, such as the NICT Japanese Learner of English corpus and the Chinese Learner English Corpus (Shichun and Huizhong, 2003), but the lack of definitive corpora often makes obtaining data for use in training models a task within itself, and often limits the approaches which can be taken.

Using classification or rule-based systems for grammatical error detection has proven to be successful to some extent, but many approaches are not sufficient for real-world automated grammar correction for various of reasons. First, as we have already mentioned, classification systems and generation-based systems do not make full use of the given data when trying to make a selection. This limits the system's ability to make well-informed edits which match the writer's original intent. Second, many of the systems start with the assumption that there is only one type of error. However, ESL students often make several combined mistakes in one sentence. These combined mistakes can throw off error detection/correction schemes which assume that the rest of the sentence is correct. For example, if a student erroneously writes 'much poeple' instead of 'many people', a system trying to correct 'many/much' errors may skip correction of much to many because it does not have any reference to the misspelled word

'poeple'. Thus there are advantages in looking at the sentence as a whole, and creating models which allow several types of errors to occur within the same sentence. We now present our model, which supports the addition of various types of errors into one combined model, and derives its response by using the whole of the observed sentence.

3 Base Model

Our noisy channel model consists of two main components, a base language model and a noise model. The base language model is a probabilistic language model which generates an 'error-free' sentence[1] with a given probability. The probabilistic noise model then takes this sentence and decides whether or not to make it erroneous by inserting various types of errors, such as spelling mistakes, article choice errors, wordform choice errors, etc., based on its parameters (see Figure 1 for example). Using this model, we can find the posterior probability $p(S_{orig}|S_{obs})$ using Bayes rule where S_{orig} is the original sentence created by our base language model, and S_{obs} is the observed erroneous sentence.

$$p(S_{orig}|S_{obs}) = \frac{p(S_{obs}|S_{orig})p(S_{orig})}{p(S_{obs})}$$

For the language model, we can use various known probabilistic models which already have defined methods for learning the parameters, such as n-gram models or PCFGs. For the noise model, we need some way to learn the parameters for the mistakes that a group of specified writers (such as Korean ESL students) make. We address this issue in Section 4.

Using this model, we can find the highest likelihood error-free sentence for an observed output sentence by tracing all possible paths from the language model through the noise model and ending in the observed sentence as output.

4 Implementation

To actually implement our model, we use a bigram model for the base language model, and various noise models which introduce spelling errors, article choice errors, preposition choice errors, etc.

[1] In reality, the language model will most likely produce sentences with errors as seen by humans, but from the modeling perspective, we assume that the language model is a perfect representation of the language for our task.

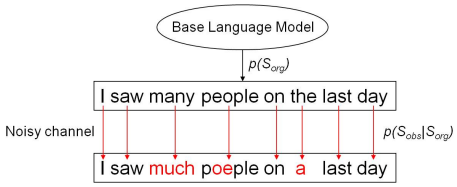

Figure 1: Example of noisy channel model

All models are implemented using weighted finite-state tranducers (wFST). For operations on the wFSTs, we use OpenFST (Allauzen et al., 2007), along with expectation semiring code supplied by Markus Dryer for Dreyer et al. (2008).

4.1 Base language model

The base language model is a bigram model implemented by using a weighted finite-state transducer (wFST). The model parameters are learned from the British National Corpus modified to use American English spellings with Kneser-Ney smoothing. To lower our memory usage, only bigrams whose words are found in the observed sentences, or are determined to be possible candidates for the correct words of the original sentence (due to the noise models) are used. While we use a bigram model here for simplicity, any probabilistic language model having a tractable intersection with wFSTs could be used. For the bigram model, each state in the wFST represents a bigram context, except the end state. The arcs of the wFST are set so that the weight is the bigram probability of the output word given the context specified by the from state, and the output word is a word of the vocabulary. Thus, given a set of n words in the vocabulary, the language model wFST had one start state, from which n arcs extended to each of their own context states. From each of these nodes, $n + 1$ arcs extend to each of the n context states and the end state. Thus the number of states in the language model is $n + 2$ and the number of arcs is $O(n^2)$.

4.2 Noise models

For our noise model, we created a weighted finite-state transducer (wFST) which accepts error-free input, and outputs erroneous sentences with a specified probability. To model various types of human errors, we created several different noise models and

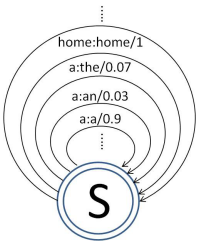

Figure 2: Example of noise model

composed them together, creating a layered noise model. The noise models we implement are spelling errors, article choice errors, preposition choice errors, and insertion errors, which we will explain in more detail later in this section.

The basic design of each noise wFST starts with an initial state, which is also the final state of the wFST. For each word found in the language model, an arc going from the initial state to itself is created, with the input and output values set as the word. These arcs model the case of no error being made. In addition to these arcs, arcs representing prediction errors are also inserted. For example, in the article choice error model, an arc is added for each possible (*input, output*) article pair, such as *a*:*an* for making the mistake of writing *an* instead of *a*. The weights of the arcs are the probabilities of introducing errors, given the input word from the language model. For example, the noise model shown in Figure 2 shows a noise model in which *a* will be written correctly with a probability of 0.9, and will be changed to *an* or *the* with probabilities 0.03 and 0.07, respectively. For this model to work correctly, the setting of the probabilities for each error is required. How this is done is explained in Section 4.3.

4.2.1 Spelling errors

The spelling error noise model accounts for spelling errors made by writers. For spelling errors, we allowed all spelling errors which were a Damerau-Levenshtein distance of 1 (Damerau, 1964; Levenshtein, 1966). While allowing a DL distance of 2 or higher may likely have better perfomance, the model was constrained to a distance of 1 due to memory constraints. We specified one parameter λ_n for each possible word length n. This parameter is the total probability of making a spelling error for a given word length. For each word length we distributed the probability of each possible spelling error equally. Thus for word length n, we have n deletion errors, $25n$ substitution errors, $n-1$ transposition errors, and $26(n+1)$ insertion errors, and the probability for each possible error is $\frac{\lambda_n}{n+25n+n-1+26(n+1)}$. We set the maximum word length for spelling errors to 22, giving us 22 parameters.

4.2.2 Article choice errors

The article choice error noise model simulates incorrect selection of articles. In this model we learn $n(n-1)$ parameters, one for each article pair. Since there are only 3 articles (*a, an, the*), we only have 6 parameters for this model.

4.2.3 Preposition choice errors

The preposition choice error noise model simulates incorrect selection of prepositions. We take the 12 most commonly misused prepositions by ESL writers (Gamon et al., 2009) and specify one parameter for each preposition pair, as we do in the article choice error noise model, giving us a total of 132 parameters.

4.2.4 Wordform choice errors

The wordform choice error noise model simulates chosing the incorrect wordform of a word. For example, choosing the incorrect tense of a verb (e.g. *went→go*), or the incorrect number marking on a noun or verb (e.g. *are→is*) would be a part of this model. This error model has one parameter for every number of possible inflections, up to a maximum of 12 inflections, giving us 12 parameters. The parameter is the total probability of choosing the wrong inflection of a word, and the probability is spread evenly between each possible inflection. We used CELEX (Baayen et al., 1995) to find all the possible wordforms of each observed word.

4.2.5 Word insertion errors

The word insertion error model simulates the addition of extraneous words to the original sentence. We create a list of words by combining the prepositions and articles found in the article choice and preposition choice errors. We assume that the words on the list have a probability of being inserted erroneously. There is a parameter for each word, which

is the probability of that word being inserted. Thus we have 15 parameters for this noise model.

4.3 Learning noise model parameters

To achieve maximum performance, we wish to learn the parameters of the noise models. If we had a large set of erroneous sentences, along witha hand-annotated list of the specific arrors and their corrections, it would be possible to do some form of supervised learning to find the parameters. We looked at the NICT Japanese Learner of English (JLE) corpus, which is a corpus of transcripts of 1,300 Japanese learners' English oral proficiency interview. This corpus has been annotated using an error tagset (Izumi et al., 2004). However, because the JLE corpus is a set of transcribed sentences, it is in a different domain from our task. The Chinese Learner English Corpus (CLEC) contains erroneous sentences which have been annotated, but the CLEC corpus had too many manual errors, such as typos, as well as many incorrect annotations, making it very difficult to automate the processing. Many of the corrections themselves were also incorrect. We were not able to find of a set of annotated errors which fit our task, nor are we aware that such a set exists. Instead, we collected a large data set of possibly erroneous sentences from Korean ESL students (Section 5.1). Since these sentences are not annotated, we need to use an unsupervised learning method to learn our parameters.

To learn the parameters of the noise models, we assume that the collected sentences are random output of our model, and train our model using the EM algorithm. This was done by making use of the V-expectation semiring (Eisner, 2002). The V-expectation semiring is a semiring in which the weight is defined as $\mathbb{R}_{\geq 0} \times V$, where $\mathbb{R}$ can be used to keep track of the probability, and V is a vector which can be used to denote arc traversal counts or feature counts. The weight for each of the arcs in the noise models was set so that the real value was the probability and the vector V denoted which choice (having a specified error or not) was made by selecting the arc. We create a generative language-noise model by composing the language model wFST with the noise model wFSTs, as shown in Figure 3. By using the expectation semiring, we can keep track of the probability of each path going over an erroneous

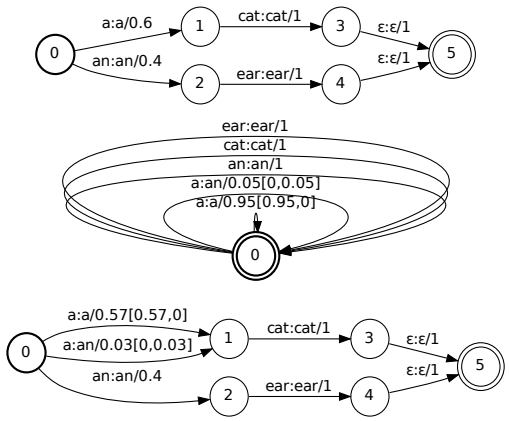

Figure 3: Example of language model (top) and noise model (middle) wFST composition. The vector of the V-expectation semiring weight is in brackets. The first value of the vector denotes no error being made on writing 'a' and the second value denotes the error of writing 'an' instead of 'a'

arc or non-erroneous arc.

Once our model is set up for the E step using the initial parameters, we must compute the expected number of noise model arc traversals for use in calculating our new parameters. To do this, we need to find all possible paths resulting in the observed sentence as output, for each observed sentence. Then, for each possible path, we need to calculate the probability of the path given the output sentence, and get the expected counts of going over each erroneous and error-free arc to learn the parameters of the noise model. To find a wFST with just the possible paths for each observed sentence, we can compose the language-noise wFST with the observed sentence wFST. The observed sentence wFST is created in the following manner. Given an observed sentence, an initial state is created. For each word in the sentence, in the order appearing in the sentence, a new state is added, and an arc is created going from the previously added state to the newly added state. The new arc takes the observed word as input and also uses it as output. The weight/probability for each arc is set to 1. Composing the sentence wFST with the language-noise wFST has the effect of restricting the new wFST to only have sentences which output the observed sentence from the language-noise wFST. We now have a new wFST where all valid paths are the paths which can produce the observed

sentence. To find the total weight of all paths, we first change all input and output symbols into the empty string. Since all arcs in this wFST are epsilon arcs, we can use the epsilon-removal operation (Mohri, 2002), which will reduce the wFST to one state with no arcs. This operation combines the total weight of all paths into the final weight of the sole state, giving us the total expectation value for that sentence. By doing this for each sentence, and adding the expectation values for each sentence, we can easily compute the expectation step, from which we can find the maximizing parameters and update our parameters accordingly.

4.4 Finding the maximum likelihood correction

Once the parameters are learned, we can use our model to find the maximum likelihood error-free sentence. This is done by again creating the language model and noise model with the learned parameters, but this time we set the weights of the noise model to just the probabilities, using the log semiring, since we do not need to keep track of expected values. We also set the language model input for each arc to be the same word as the output, instead of using an empty string. Once again, we compose the language model with the noise models. We create a sentence wFST using the observed sentence we wish to correct, the same way the observed sentence wFST for training was created. This is now composed with the language-noise wFST. Now all we need to do is find the shortest path (when using minus-log probabilities) of the new wFST, and the input to that path will be our corrected sentence.

5 Experiment

We now present the data set and evaluation technique used for our experiments.

5.1 Data Set

To train our noise models, we collected around 25,000 essays comprised of 478,350 sentences written by Korean ESL students preparing for the TOEFL writing exam. These were collected from open web postings by Korean ESL students asking for advice on their writing samples. In order to automate the process, a program was written to download the posts, and discard the posts that were deemed too short to be TOEFL writing samples. Also discarded were the posts that had a "[re" or "re.." in the title. Next, all sentences containing Korean were removed, after which some characters were changed so that they were in ASCII form. The remaining text was separated into sentences solely by punctuation marks ., *!*, and *?*. This resulted in the 478,350 sentences stated above. Due to the process, some of the sentences collected are actually sentence fragments, where punctuation had been misused. For training and evaluation purposes, the data set was split into a test set with 504 randomly selected sentences, an evaluation set of 1017 randomly selected sentences, and a training set composed of the remaining sentences.

5.2 Evaluation technique

In the current literature, grammar correction tasks are often manually evaluated for each output correction, or evaluated by taking a set of proper sentences, artificially introducing some error, and seeing how well the algorithm fixes the error. Manual evaluation of automatic corrections may be the best method for getting a more detailed evaluation, but to do manual evaluation for every test output requires a large amount of human resources, in terms of both time and effort. In the case where artificial lesioning is introduced, the lesions may not always reflect the actual errors found in human data, and it is difficult to replicate the actual tendency of humans to make a variety of different mistakes in a single sentence. Thus, this method of evaluation, which may be suitable for evaluating the correction performance of specific grammatical errors, would not be fit for evaluating our model's overall performance. For evaluation of the given task, we have incorporated evaluation techniques based on current evaluation techniques used in machine translation, BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007).

Machine translation addresses the problem of changing a sentence in one language to a sentence of another. The task of correcting erroneous sentences can also be thought of as translating a sentence from a given language A, to another language B, where A is a broken language, and B is the correct language. Under this context, we can apply machine translation evaluation techniques to evaluate the performance of our system. Our model's sentence correc-

tions can be thought of as the output translation to be evaluated. In order to use BLEU and METEOR, we need to have reference translations on which to score our output. As we have already explained in section 5.1, we have a collection of erroneous sentences, but no corrections. To obtain manually corrected sentences for evaluation, the test and evaluation set sentences and were put on Amazon Mechanical Turk as a correction task. Workers residing in the US were asked to manually correct the sentences in the two sets. Workers had a choice of selecting 'Impossible to understand', 'Correct sentence', or 'Incorrect sentence', and were asked to correct the sentences so no spelling errors, grammatical errors, or punctuation errors were present. Each sentence was given to 8 workers, giving us a set of 8 or fewer corrected sentences for each erroneous sentence. We asked workers not to completely rewrite the sentences, but to maintain the original structure as much as possible. Each hit was comprised of 6 sentences, and the reward for each hit was 10 cents. To ensure the quality of our manually corrected sentences, a native English speaker research assistant went over each of the 'corrected' sentences and marked them as correct or incorrect. We then removed all the incorrect 'corrections'.

Using our manually corrected reference sentences, we evaluate our model's correction performance using METEOR and BLEU. Since METEOR and BLEU are fully automated after we have our reference translations (manual corrections), we can run evaluation on our tests without any need for further manual input. While these two evaluation methods were created for machine translation, they also have the potential of being used in the field of grammar correction evaluation. One difference between machine translation and our task is that finding the right lemma is in itself something to be rewarded in MT, but is not sufficient for our task. In this respect, evaluation of grammar correction should be more strict. Thus, for METEOR, we used the 'exact' module for evaluation.

To validate our evaluation method, we ran a simple test by calculating the METEOR and BLEU scores for the observed sentences, and compared them with the scores for the manually corrected sentences, to test for an expected increase. The scores for each correction were evaluated using the set of corrected sentences minus the correction sentence being evaluated. For example, let us say we have the observed sentence o, and correction sentences c_1, c_2, c_3 and c_4 from Mechanical Turk. We run METEOR and BLEU on both o and c_1 using c_2, c_3 and c_4 as the reference set. We repeat the process for o and c_2, using c_1, c_3 and c_4 as the reference, and so on, until we have run METEOR and BLEU on all 4 correction sentences. With a set of 100 manually labeled sentences, the average METEOR score for the ESL sentences was 0.8327, whereas the corrected sentences had an average score of 0.9179. For BLEU, the average scores were 0.7540 and 0.8786, respectively, as shown in Table 1. Thus, we have confirmed that the corrected sentences score higher than the ESL sentence. It is also notable that finding corrections for the sentences is a much easier task than finding various correct translations, since the task of editing is much easier and can be done by a much larger set of qualified people.

	METEOR	BLEU
Original ESL sentences	0.8327	0.7540
Manual corrections	0.9179	0.8786

Table 1: BLEU and METEOR scores for ESL sentences vs manual corrections on 100 randomly chosen sentences

	METEOR	BLEU
Aspell	0.824144	0.719713
Spelling noise model	0.825001	0.722383

Table 2: Aspell vs Spelling noise model

6 Results

For our experiments, we used 2000 randomly selected sentences for training, and a set of 1017 annotated sentences for evaluation. We also set aside a set of 504 annotated sentences as a development set. With the 2000 sentence training, the performance generally converged after around 10 iterations of EM.

6.1 Comparison with Aspell

To check how well our spelling error noise model is doing, we compared the results of using the spelling error noise model with the output results of using the GNU Aspell 0.60.6 spelling checker. Since we

	METEOR	↑	↓	BLEU	↑	↓
ESL Baseline	0.821000			0.715634		
Spelling only	0.825001	49	5	0.722383	53	8
Spelling, Article	0.825437	55	6	0.723022	59	9
Spelling, Preposition	0.824157	52	17	0.720702	55	19
Spelling, Wordform	0.825654	81	25	0.723599	85	27
Spelling, Insertion	0.825041	52	5	0.722564	56	8

Table 3: Average evaluation scores for various noise models run on 1017 sentences, along with counts of sentences with increased (↑) and decreased (↓) scores. All improvements are significant by the binomial test at $p < 0.001$

are using METEOR and BLEU for our evaluation metric, we needed to get a set of corrected sentences for using Aspell. Aspell lists the suggested spelling corrections of misspelled words in a ranked order, so we replaced each misspelled word found by Aspell with the word with the highest rank (lowest score) for the Aspell corrections. One difference between Aspell and our model is that Aspell only corrects words which do not appear in the dictionary, while our method looks at all words, even those found in the dictionary. Thus our model can correct words which look correct by themselves, but seem to be incorrect due to the bigram context. Another difference is that Aspell has the capability to split words, whereas our model does not allow the insertion of spaces. A comparison of the scores is shown in Table 2. We can see that our model has better performance, due to better word selection, despite the advantage that Aspell has by using phonological information to find the correct word, and the disadvantage that our model is restricted to spellings which are within a Damerau-Levenstein distance of 1. This is due to the fact that our model is context-sensitive, and can use other information in addition to the misspelled word. For example, the sentence 'In contast, high prices of products would be the main reason for dislike.' was edited in Aspell by changing 'contast' to 'contest', while our model correctly selected 'contrast'. The sentence 'So i can reach the theater in ten minuets by foot' was not edited by Aspell, but our model changed 'minuets' to 'minutes'. Another difference that can be seen by looking through the results is that Aspell changes every word not found in the dictionary, while our algorithm allows words it has not seen by treating them as unknown tokens. Since we are using smoothing, these tokens are left in place if there is no other high probability bigram to take its place. This helps leave intact the proper nouns and words not in the vocabulary.

6.2 Noise model performance and output

Our next experiment was to test the performance of our model on various types of errors. Table 3 shows the BLEU and METEOR scores of our various error models, along with the number of sentences achieving improved and reduced scores. As we have already seen in section 6.1, the spelling error model increases the evaluation scores from the ESL baseline. Adding in the article choice error model and the word insertion error models in addition to the spelling error noise model increases the BLEU score performance of finding corrections. Upon observing the outputs of the corrections on the development set, we found that the corrections changing *a* to *an* were all correct. Changes between *a* and *the* were sometimes correct, and sometimes incorrect. For example, 'which makes me know a existence about' was changed to 'which makes me know *the* existence about', 'when I am in a trouble.' was changed to 'when I am in *the* trouble.', and 'many people could read a nonfiction books' was changed to 'many people could read *the* nonfiction books'. For the last correction, the manual corrections all changed the sentence to contain 'many people could read a nonfiction book', bringing down the evaluation score. Overall, the article corrections which were being made seemed to change the sentence for the better, or left it at the same quality.

The preposition choice error model decreased the performance of the system overall. Looking through the development set corrections, we found that many correct prepositions were being changed to incorrect prepositions. For example, in the sentence 'Distrust about desire between two have been growing in their

relationship.', *about* was changed to *of*, and in 'As time goes by, ...', *by* was changed to *on*. Since these changes were not found in the manual corrections, the scores were decreased.

For wordform errors, the BLEU and METEOR scores both increased. While the wordform choice noise model had the most sentences with increased scores, it also had the most sentences with decreased scores. Overall, it seems that to correct wordform errors, more context than just the preceding and following word are needed. For example, in the sentence 'There are a lot of a hundred dollar phones in the market.', *phones* was changed to *phone*. To infer which is correct, you would have to have access to the previous context 'a lot of'. Another example is '..., I prefer being indoors to going outside ...', where *going* was changed to *go*. These types of cases illustrate the restrictions of using a bigram model as the base language model.

The word insertion error model was restricted to articles and 12 prepositions, and thus did not make many changes, but was correct when it did. One thing to note is that since we are using a bigram model for the language model, the model itself is biased towards shorter sentences. Since we only included words which were needed when they were used, we did not run into problems with this bias. When we tried including a large set of commonly used words, we found that many of the words were being erased because of the bigrams models probabilistic preference for shorter sentences.

6.3 Limitations of the bigram language model

Browsing through the development set data, we found that many of our model's incorrect 'corrections' were the result of using a bigram model as our language model. For example, '.., I prefer being indoors to going outside in that...' was changed to '.., I prefer being indoors to *go* outside in that...'. From the bigram model, the probabilities $p(go \mid to)$ and $p(outside|go)$ are both higher than $p(going|to)$ and $p(outside \mid going)$, respectively. To infer that going is actually correct, we would need to know the previous context, that we are comparing 'being indoors' to 'going outside'. Unfortunately, since we are using a bigram model, this is not possible. These kind of errors are found throughout the corrections. It seems likely that making use of a language model which can keep track of this kind of information would increase the performance of the correction model by preventing these kinds of errors.

7 Conclusion and future work

We have introduced a novel way of finding grammar and spelling corrections, which uses the EM algorithm to train the parameters of our noisy channel approach. One of the benefits of this approach is that it does not require a parallel set of erroneous sentences and their corrections. Also, our model is not confined to a specific error, and various error models may be added on. For training our noise model, all that is required is finding erroneous data sets. Depending on which domain you are training on, this can also be quite feasible as we have shown by our collection of Korean ESL students' erroneous writing samples. Our data set could have been for ESL students of any native language, or could also be a data set of other groups such as young native English speakers, or the whole set of English speakers for grammar correction. Using only these data sets, we can train our noisy channel model, as we have shown using a bigram language model, and a wFST for our noise model. We have also shown how to use weighted finite-state transducers and the expectation semiring, as well as wFST algorithms implemented in OpenFST to train the model using EM. For evaluation, we have introduced a novel way of evaluating grammar corrections, using MT evaluation methods, which we have not seen in other grammar correction literature. The produced corrections show the restrictions of using a bigram language model. For future work, we plan to use a more accurate language model, and add more types of complex error models, such as word deletion and word ordering error models to improve performance and address other types of errors.

Acknowledgments

We are grateful to Randy West for his input and assistance, and to Markus Dreyer who provided us with his expectation semiring code. We would also like to thank the San Diego Supercomputer Center for use of their DASH high-performance computing system.

References

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer. `http://www.openfst.org`.

Baayen, H. R., Piepenbrock, R., and Gulikers, L. (1995). *The CELEX Lexical Database. Release 2 (CD-ROM)*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, Pennsylvania.

Brockett, C., Dolan, W. B., and Gamon, M. (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*, ACL-44, pages 249–256, Morristown, NJ, USA. Association for Computational Linguistics.

Chodorow, M. and Leacock, C. (2000). An unsupervised method for detecting grammatical errors. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 140–147, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7:171–176.

De Felice, R. and Pulman, S. G. (2008). A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 169–176, Morristown, NJ, USA. Association for Computational Linguistics.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38.

Désilets, A. and Hermet, M. (2009). Using automatic roundtrip translation to repair general errors in second language writing. In *Proceedings of the twelfth Machine Translation Summit*, MT Summit XII, pages 198–206.

Dreyer, M., Smith, J., and Eisner, J. (2008). Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1080–1089, Honolulu, Hawaii. Association for Computational Linguistics.

Eeg-olofsson, J. and Knutsson, O. (2003). Automatic grammar checking for second language learners - the use of prepositions. In *In Nodalida*.

Eisner, J. (2002). Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1–8, Philadelphia.

Gamon, M., Leacock, C., Brockett, C., Dolan, W. B., Gao, J., Belenko, D., and Klementiev, A. (2009). Using statistical techniques and web search to correct ESL errors. In *Calico Journal*, Vol 26, No. 3, pages 491–511, Menlo Park, CA, USA. CALICO Journal.

Izumi, E., Uchimoto, K., and Isahara, H. (2004). The NICT JLE corpus exploiting the language learnersspeech database for research and education. In *International Journal of the Computer, the Internet and Management*, volume 12(2), pages 119–125.

Izumi, E., Uchimoto, K., Saiga, T., Supnithi, T., and Isahara, H. (2003). Automatic error detection in the Japanese learners' English spoken data. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, ACL '03, pages 145–148, Morristown, NJ, USA. Association for Computational Linguistics.

Knight, K. and Chander, I. (1994). Automated postediting of documents. In *Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, AAAI '94, pages 779–784, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Lavie, A. and Agarwal, A. (2007). Meteor: an automatic metric for MT evaluation with high levels of correlation with human judgments. In *StatMT '07: Proceedings of the Second Workshop on*

Statistical Machine Translation, pages 228–231, Morristown, NJ, USA. Association for Computational Linguistics.

Lee, J. and Seneff, S. (2006). Automatic grammar correction for second-language learners. In *Proceedings of Interspeech*.

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

Mohri, M. (2002). Generic epsilon-removal and input epsilon-normalization algorithms for weighted transducers. In *International Journal of Foundations of Computer Science 13*, pages 129–143.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Shannon, C. (1948). A mathematical theory of communications. *Bell Systems Technical Journal*, 27(4):623–656.

Shichun, G. and Huizhong, Y. (2003). Chinese Learner English Corpus. Shanghai Foreign Language Education Press.

Tetreault, J. R. and Chodorow, M. (2008). The ups and downs of preposition error detection in ESL writing. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 865–872, Morristown, NJ, USA. Association for Computational Linguistics.

Wu, C.-F. J. (1983). On the convergence properties of the EM algorithm. *Ann. Statist.*, 11(1):95–103.