TellMeWhy: Learning to Explain Corrective Feedback for Second Language Learners

Yi-Huei Lai, Jason Chang
National Tsing Hua University, Hsinchu, Taiwan
{yima, jason}@nlplab.cc

Abstract

We present a writing prototype feedback system, *TellMeWhy*, to provide explanations of errors in submitted essays. In our approach, the sentence with corrections is analyzed to identify error types and problem words, aimed at customizing explanations based on the context of the error. The method involves learning the relation of errors and problem words, generating common feedback patterns, and extracting grammar patterns, collocations and example sentences. At run-time, a sentence with corrections is classified, and the problem word and template are identified to provide detailed explanations. Preliminary evaluation shows that the method has potential to improve existing commercial writing services.

1 Introduction

Many English essays and sentences with grammatical errors (e.g., 'We discussed about the issue.') are submitted by L2 learners to writing services every day, and an increasing number of online grammar checkers specifically target learners' essays. For example, *Write & Improve* (writeandimprove.com) pinpoints an error without explaining the correction. *Grammarly* (www.grammarly.com) corrects grammatical errors and provides context-insensitive examples using pre-compiled information.

Writing services such as *Write & Improve* and *Grammarly* typically use canned text to explain a common error. However, corrective feedback with the most useful and comprehensive explanations may contain collocations, grammar, and context-sensitive examples. These systems could provide better explanations to the user, if the context of the correction is taken into consideration.

Consider the sentence '*We discussed about the issue.*' The best explanation for the error is probably not (1) and (2) which only state the obvious

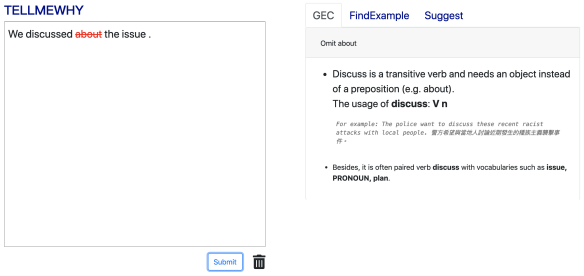

Figure 1: Example **TellMeWhy-GEC** for the error 'We discussed about the issue.'

(i.e. editing operations for a correction), but rather (3) which explains using the grammar patterns of the problem word (i.e., *discuss*) with a simple example.

(1) This word *about* may not be needed. Is your writing clearer without it? (from *Write & Improve*)

(2) The preposition seems unnecessary after the verb discussed. Consider removing the preposition. (from *Grammarly*)

Examples:

✗ The government advocated to recycling.

✓ The government advocated recycling.

(3) **discuss sth** (WITHOUT **about/on**): 'He simply refuses to discuss the matter.'

Compare **talk about, a discussion about/on**: 'They want to talk about what to do next.' (from *Longman Common Error Dictionary (Turton and Heaton, 1996)*)

We present a prototype system, *TellMeWhy*, with three functions: *GEC* (in Figure 1), *FindExample*, and *Suggest*. *Suggest* displays statistical data of erroneous and corrected usage to explain correctness; *GEC* and *FindExample* explain corrections using linguistic characteristics. With an underlying grammatical error correction (GEC) system, we will provide feedback based on the correction made by the GEC. However, if the GEC system can not detect an error, our explanation module cannot be activated. For this, *FindExample* can search for a sentence with correction edits

Proceedings of the 2019 EMNLP and the 9th IJCNLP (System Demonstrations), pages 235–240
Hong Kong, China, November 3 – 7, 2019.

and display explanations so *TellMeWhy* will not being limited by the underlying GEC system.

At run-time, *TellMeWhy* starts with a given sentence, potentially with corrections made by the underlying GEC system or submitted to *FindExample*. *TellMeWhy* then generates feedback after identifying error types, problem words, and linguistic information as described in Section 3.1. We develop explanation templates based on handcrafted training data. In our prototype, *TellMeWhy* returns the feedback to the end user directly (Figure 1); also, the feedback can be shown to a human rater in order to assist them in rating essays.

2 Related Work

Corrective feedback on learners' writing has been an area of active research. Recently, the state-of-the-art in the research has been represented in Leacock et al. (2010) involving many automated approaches to detect a wide range of error types.

To generate explanations, most writing services adopt the approach of canned text with table look-up schemes (Andersen et al., 2013). This approach, despite providing basic explanations, is not optimal, because canned feedback tends to be superficial and context-insensitive. Additionally, canned feedback is limited to common grammatical errors, without covering lexical error types, such as word-choice errors.

Providing corrective feedback and explaining how the correction improves grammaticality and adheres to idiomatic principles (Sinclair, 1991) have long been an important research area in Teaching English as Second Language (TESOL). Bitchener et al. (2005) concluded that proper error feedback is significant in improving English writing. In this work, we focus on identifying the problem-causing word and providing a comprehensive and context-sensitive explanation based on the problem word.

In a study more closely related to our work, Nicholls (2003) describes an error coding scheme for lexicography and English Language Teaching (ELT) research based on a Cambridge Learner Corpus. Turton and Heaton (1996) compiled examples of common errors organized by problem words based on Longman Learners' Corpus. Chen et al. (2017) describes methods for providing explanations based on grammar patterns.

We used the data in Turton and Heaton (1996) as the main source of training data and adopted the explaining strategy proposed by Chen et al. (2017). The main difference between our work and Chen et al. (2017) is that Chen et al. (2017) provided limited error types with grammar patterns as explanation, while our method provides context-sensitive explanations of more error types.

(1) Classifying the error type
(2) Extracting grammar patterns, collocations as well as examples and definitions
(3) Calculating co-occurrence frequency of pairs of an edit and neighboring words
(4) Building explanation templates based on handcrafted training data

Figure 2: Outline of the process used to train the *TellMeWhy* system

In contrast to the existing systems on corrective feedback, *TellMeWhy* provides comprehensive explanations for multiple error types with an underlying GEC system. Additionally, to provide good explanations, a promising approach is to automatically classify the error and extract a problem word nearby to tailor the explanation to the context of the error.

3 The *TellMeWhy* System

We focus on providing comprehensive feedback on a given English sentence with corrections. The feedback is returned as an output to the end user directly. It is important to detect the potential problem-causing word and classify the error. Our goal is returning a context-sensitive explanation using the problem word and the error type to instantiate suitable explanation template and fetch reference linguistic information.

3.1 Learning to Provide Feedback

We attempt to learn to generate an informative and comprehensive explanation that matches the error and context (in particular, the problem word). Our learning process is shown in Figure 2.

In the first stage of the learning process (Step (1) in Figure 2), we analyze sentences with a correction annotated with a problem-causing word which is a word regularly causing the specific error. As we will describe in Section 3.1, we use Turton and Heaton (1996) for training. Based on an existing method, ERRANT (Bryant et al., 2017), we analyze differences between a sentence containing an error and a corrected sentence. Then we produce an error type including an edit type (insert, delete, and replace) and PoS of the edit

Table 1: top 10 error codes

Code	Gloss	Samples Sentence
1. SP	Spelling	[-Fortunaly-]{+Fortunately+}, the police found her.
2. RV	Replace v.	How to [-make-]{+create+} a better hairstyle.
3. RT	Replace prep.	We should invest more money [-to-]{+in+} education.
4. MD	Missing det.	School finishes at five in {+the+} afternoon.
5. R	Replace w.	Some people tried to enter without [-any-]{+a+} ticket.
6. RN	Replace n.	Television provides many [-advantages-]{+benefits+}.
7. FV	Form v.	The problems have been [-arised-]{+arisen+} due to overpopulation.
8. MT	Missing prep.	He apologized {+to+} her for the long delay.
9. UD	Useless det.	I have work to do. I don't have [-a-] time for anything else.
10. UT	Useless prep.	We discussed [-about-] the issue.

(1) Generating a set of phrase templates in the form of PoS tags
(2) Extracting grammar patterns for all keywords in the given corpus annotated by PoS based on phrase templates
(3) Extracting exemplary instances for all patterns of all keywords

Figure 3: Outline of the pattern extraction process

(e.g., *(DEL, PREP, about)*). For simplicity, we limit ourselves to Top *10* most common error types (in Table 1) in CLE-FCE (Yannakoudakis et al., 2011) and associated error types (i.e., extending error types from Top *10*, such as from replacing a word, to deleting a word or missing a word) to derive explanation templates. To sum up, we limit ourselves to provide explanation for error types related to editing a verb, adjective, noun, prepositions and function words (e.g., articles).

In the second stage of the learning algorithm (Step (2) in Figure 2), we extract reference information. First, we extract grammar patterns from Corpus of Contemporary American English (*COCA*[1]) using an existing method (in Figure 3) described in Chang and Chang (2015). Subsequently, we store examples corresponded to a keyword's grammar patterns (e.g., (discuss, V n): issue, topic, matters). Next, we build a collocation dictionary with dependency relation information using triples (e.g., (eat, V:obj:N, bananas)). The relation (e.g., V:obj:N) is produced by a dependency parser using *COCA*. Finally, we store definitions in *Online Cambridge Dictionary* to explain word-choice errors (e.g., 'accept' education). Additionally, *Online Cambridge Dictionary* gives a pair of English-Chinese definition and a guide word (e.g., TAKE, APPROVE) for each sense of a polysemous word (e.g., accept). The guide words are instrumental for us to find the closest sense between erroneous and correction words.

In the third stage of the learning algorithm (Step (3) in Figure 2), we calculate co-occurrence frequency of pairs of an edit and neighboring

[1]www.english-corpora.org/coca

(1) Generating skipped bigrams from a large corpus and storing corresponded distance
(2) Filtering the collocates
(3) Filtering the distances for each collocates

Figure 4: Outline of the Smadja's process

words using the EF-Cambridge Open Language Database *EFCAMDAT* (Geertzen et al. (2013) and Huang et al. (2018)). Then, we use the method proposed by Smadja (1993) (in Figure 4) to calculate co-occurrence frequency of pairs of an edit and neighboring words with the goal of selecting the most potential neighboring word triggering the edit. In other words, we assume the problem-causing word as a collocate of the edit.

In the fourth and final stage of training (Step (4) in Figure 2), we formulate a feedback template for each error type, classified by *ERRANT*. For each error type, we observe and exploit the consistent patterns of feedback in Turton and Heaton (1996) to design the templates. For example, we formulate an explanation template for unnecessary preposition error with three components: problem word, grammar pattern, and example. An example feedback template is shown in Table 2. After inferencing from explanation instances of each error type, we develop type-specific templates with slots to be filled with specific information: problem words along with a grammar pattern, collocations and examples.

3.2 Run-Time Feedback

Once feedback templates have been designed and reference information for problem words has been extracted, they are stored in tables.

At run-time, users submit an essay possibly with some errors (e.g., *We discussed about the issue*). The underlying GEC system corrects the essay with edits (e.g., We discussed [-about-] the issue). Next, *TellMeWhy* determines the error type (e.g., (*DEL*,*PREP*,*about*)) and the problem word (e.g., discussed). We then produce as follows.

First, we handle corrections in the input sentence one by one. We correct an input sentence with multiple corrections to be only one correction edit temporally and iteratively. After recording the correction information (e.g., the editing type, erroneous and correction word), we also correct the only one correction because correction tags could influence a tagger to tag each word with PoS. Annotated the input sentence with PoS, we could ex-

Table 2: Sample correction-feedback pairs from the training collection

Problem Word	Sentence and Feedback	Analysis
discuss	✗ They would like to discuss about what to do next. ✓ They would like to discuss what to do next. **discuss sth** (WITHOUT **about/on**): He simply refuses to discuss the matter. There is nothing to discuss.	$pw = 'discuss' $edits = (DEL, PREP, about) $gp = V n
Template	**feedback**($pw,$gp,$edits) = $pw $gp (WITHOUT $edits[2]): $example($pw,$gp)	

[1] $gp denotes a grammar pattern.
[2] $pw denotes a problem-trigger word.
[3] $example denotes an example consisting a grammar pattern based on a problem word.

Table 3: Template for Replace a Preposition

Problem Word	Sentence and Feedback	Analysis
arm	✗ She would not stop crying until I held her on my arms. ✓ She would not stop crying until I held her in my arms. **(hold sb/sth) in your arms** (NOT on): He had a great pile of books in his arms.	$pw = arm $edits = (RP, PREP, on, in) $gp = in N
Template	**feedback**($pw,$gp,$edits) = $gp (NOT $edits[2]): $example($pw,$gp)	

tract grammar patterns in the input sentence covering correction using the method in Figure 3.

Next, to make feedback relevant to the context, we identify the most potential problem verb, noun or adjective through the highest co-occurrence frequency of the correction and potential problem words. The problem word is then used to instantiate the explanation template and fetch reference information of grammar patterns, collocations, examples, and problem word definitions.

4 Experiment and Evaluation

TellMeWhy was designed to generate explanations containing grammar, definitions, collocations, and examples. Thus, *TellMeWhy* was trained by using the *Longman Dictionary Of Common Errors* (Turton and Heaton, 1996) as the main source of knowledge. In this section, we first present the details of training. Next, we describe common error types tested. Finally, we introduce feedback strategy and evaluate the experimental results.

4.1 Training *TellMeWhy*

We used 70% of common errors and explanations in Turton and Heaton (1996) to train *TellMeWhy*. Turton and Heaton (1996) contains a collection of approximately 2,500 common errors and explanations. Table 2 shows one sample correction-explanation pairs with a problem word and an explanation template. Additionally, we used *EFCAMDAT*, a corpus of real learners' writings with edits by human graders and containing over 83 million words from 1 million essays written by 174,000 learners, to compute co-occurrence frequency of edits and problem words.

By analyzing a sentence with errors and their correction pairs, we produced error types including three edits: *replace*, *insert*, and *delete*, and PoS of the erroneous and correction words. We found that explaining strategies are highly related to error type. Therefore, we customized explaining strategies for each error type.

For errors related to replacing a preposition. First, we use a template (in Table 3) that shows the grammar patterns of the problem-causing word. The potential problem words are the closest noun, verb, or adjective to the erroneous preposition such as 'held' and 'arms'. We detected grammar patterns (in Figure 3) for each potential problem word (e.g., hold: V in n, arm: in N). The most relevant grammar pattern is selected by highest co-occurrence frequency of a problem word and the edit calculated in advance using *EFCAMDAT* through the method proposed by Smadja (1993). Then, we ranked collocations of an edit-problem-word pair (e.g., ([-on-]{+in+}, arm), (hold, [-on-]{+in+})) by highest co-occurrence frequency of a problem word and the edit.

As for errors related to replacing function words (e.g., articles and demonstratives), which form a closed set, we exploited feedback in Turton and Heaton (1996) and rules in Cobuild et al. (2005). Nevertheless, determiner errors related to time (e.g., in 'the' morning and at night) are difficult to handle using general determiner rules. In that case, we followed Turton and Heaton (1996) time-specific rules.

With respect to cases of replacing open-class words (i.e., verb, noun, and adjective), we handled the errors by cases: (1) spelling, (2) tense,

Table 4: Template for Replacing a Word

Problem Word	Sentence and Feedback	Analysis
abandon	✗ Since capital punishment was abandoned, the crime rate has increased. ✓ Since capital punishment was abolished, the crime rate has increased. **abandon** = give up a **plan, activity** or attempt to do something, without being successful: Bad weather forced them to abandon the search. Without government support, the project will have to be abandoned.	$pw = abandon $edits = (RP, VERB, abandon, abolish) $col[0] = [plan, activity]
Template	**feedback**($edits,$col) = $edits[2] = $def($edits[2]) and usually paired with $col[0] : $example($edits[2]) $edits[3] = $def($edits[3]) and usually paired with $col[1] : $example($edits[3])	

[1] $def denotes a definition of a word.
[2] $col denotes a collocation of $pw

Table 5: Scores (0-2) for explanations on top 10 error codes, generated by four systems: *TellMeWhy-GEC* (TMW-GEC),*TellMeWhy-FindExample* (TMW-FE) *Grammarly* (GL) and *Write & Improve* (W&I)

Code	Gloss	TMW-GEC	TMW-FE	GL	W&I
1. SP	Spelling	2	2	2	2
2. RV	Replace v.		1.3		
3. RT	Replace prep.	2	1.8	1	1
4. MD	Missing det.	2	1.8	1	1
5. R	Replace w.	1.67	1.8	1.71	1
6. RN	Replace n.		1.6		1
7. FV	Form v.	0.8	1	1.4	1
8. MT	Missing prep.	2	1.6	0.83	
9. UD	Useless det.	1	0.9	2	1
10. UT	Useless prep.	1.67	1.5	1	1
	Average score	1.64	1.53	1.37	1.13

Table 6: The number of testcases can be corrected by three systems: *TellMeWhy-GEC* (TMW-GEC), *Grammarly* (GL) and *Write & Improve* (W&I)

Code	Gloss	TMW-GEC	GL	W&I
1. SP	Spelling	10	10	10
2. RV	Replace v.	0	0	0
3. RT	Replace prep.	4	6	1
4. MD	Missing det.	5	6	1
5. R	Replace w.	6	7	5
6. RN	Replace n.	0	0	1
7. FV	Form v.	7	8	3
8. MT	Missing prep.	4	6	0
9. UD	Useless det.	7	9	3
10. UT	Useless prep.	9	8	3

(3) word choice errors. Spelling and tense errors associate with morphological variation and could be detected directly and explained easily as such.

A word-choice example pair of correction and feedback from Turton and Heaton (1996) is shown in Table 4. From this, we found word choice mistake is often made because users do not understand the definition of erroneous and corrected words very well. With this in mind, we first used the definitions from *Online Cambridge Dictionary* for the erroneous and corrected words (e.g., 'abandon' vs 'abolish'). To determine the contextually appropriate senses for polysemous words, we replaced erroneous and corrected words with all possible guide words (e.g., 'abandon': leave, stop; 'abolish': abolish) in *Online Cambridge Dictionary*. Second, we calculated cosine similarity between pairs of error-correction (e.g., leave-abolish and stop-abolish) using *Word2vec*. We choose the pair of senses with the highest cosine similarity to display definitions. Additionally, word-choice errors could be caused by miscollocation. We also provided related collocations with frequency to explain why the correction word is more appropriate.

4.2 Evaluation

Once we have trained *TellMeWhy* as described, we evaluated the performance using ten randomly-selected sentences for each top 10 common error type. We also evaluated problem word detection performance using Turton and Heaton (1996).

Two functions in our systems are used to evaluate explanations: *TellMeWhy-GEC* and *TellMeWhy-FindExample*. The former has an underlying GEC system. The latter is a writing examples search engine so that users can submit a sentence with known correction edits to understand why the edit makes sense; additionally, we can evaluate performance without being limited by the underlying GEC system.

One evaluation of comparison between *TellMeWhy* and other commercial systems is shown in Table 5 was evaluated by ten sentences for each error type and carried out by a linguist. A wrong explanation (such as a wrong problem word, inappropriate corrective feedback) gets zero point, while a correct explanation (i.e., identifying the problem word correctly but providing context-insensitive examples) gets one point. Explanations related to a problem word and context receive two points. However, the score of feedback on errors that *TellMeWhy-GEC*, *Grammarly*, and *Write & Improve* cannot detect does not count into the average score; besides, the correction performance of each system is shown in Table 6. The evaluation results show that *TellMeWhy* is considerably better

than the existing services and able to explain more error types, such as word-choice and collocation errors, and provides context-sensitive information.

The other evaluation is the performance of problem word identification. We evaluated problem word identification using Turton and Heaton (1996) containing approximately 2,500 edit-feedback pairs. We evaluated *TellMeWhy* using the rest 30% of the dataset (i.e., 750 edit-feedback pairs). Additionally, we limited ourselves to evaluate for *Top 10* error types and extension types defined in the first stage of Section 3.1. Those types of test data account for 616 editing sentences out of 750 edit-feedback pairs. For our evaluation, we treated *keywords* organized by Turton and Heaton (1996) as ground truth. The accuracy of problem word identification is approximately 80% (493/616).

5 Future Work and Summary

In this paper, we have described a system for learning to provide corrective feedback on English sentences with corrections. The method involves classifying errors into different error types, identifying potential problem words, selecting closest senses between misuse and corrected words, and extracting collocations as well as grammar patterns. We have implemented and evaluated the method as applied to real sentences. In preliminary evaluation, we have shown that the method outperforms the existing commercial systems for many error types, especially an error triggered by a verb or adjective.

Many avenues exist for future research and improvement of our system. For example, we could extend our method to handle more error types, such as sentence patterns (e.g., One ... Another ... The other, so ... that). Collocation knowledge and selectional preference could be used to improve the explanation of word-choice error types so that explanations may contain word concepts collocating with a problem word. Additionally, better methods such as learning-based approach to identifying problem words could be implemented.

References

Øistein E Andersen, Helen Yannakoudakis, Fiona Barker, and Tim Parish. 2013. Developing and testing a self-assessment and tutoring system. In *Proceedings of the eighth workshop on innovative use of NLP for building educational applications*, pages 32–41.

John Bitchener, Stuart Young, and Denise Cameron. 2005. The effect of different types of corrective feedback on esl student writing. *Journal of second language writing*, 14(3):191–205.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 793–805.

Jim Chang and Jason Chang. 2015. Writeahead2: Mining lexical grammar patterns for assisted writing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 106–110.

Jhih-Jie Chen, Jim Chang, Ching-Yu Yang, Mei-Hua Chen, and Jason S Chang. 2017. Extracting formulaic expressions and grammar and edit patterns to assist academic writing. *EUROPHRAS 2017: Computational and Corpus-based Phraseology: Recent Advances and Interdisciplinary Approaches*.

Collins Cobuild et al. 2005. *Collins Cobuild English Grammar*. Collins Cobuild.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic linguistic annotation of large scale l2 databases: The ef-cambridge open language database (efcamdat). In *Proceedings of the 31st Second Language Research Forum. Somerville, MA: Cascadilla Proceedings Project*.

Yan Huang, Akira Murakami, Theodora Alexopoulou, and Anna Korhonen. 2018. Dependency parsing of learner english. *International Journal of Corpus Linguistics*, 23(1):28–54.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. Automated grammatical error detection for language learners. *Synthesis lectures on human language technologies*, 3(1):1–134.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581.

John Sinclair. 1991. *Corpus, concordance, collocation*. Oxford University Press.

Frank Smadja. 1993. Retrieving collocations from text: Xtract. *Computational linguistics*, 19(1):143–177.

ND Turton and JB Heaton. 1996. Longman dictionary of common errors. new edition.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 180–189. Association for Computational Linguistics.