WriteAhead: Mining Grammar Patterns in Corpora for Assisted Writing

Tzu-Hsi Yen, Jian-Cheng Wu+
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan, R.O.C. 30013
{joe, jiancheng}@nlplab.cc

Joanne Boisson, Jim Chang, Jason Chang
+National Academy for Educational Research
Minstry of Education
Taipei, Taiwan, R.O.C. 30013
{joanne,jim,jason}@nlplab.cc

Abstract

This paper describes *WriteAhead*, a resource-rich, Interactive Writing Environment that provides L2 learners with writing prompts, as well as "get it right" advice, to helps them write fluently and accurately. The method involves automatically analyzing reference and learner corpora, extracting grammar patterns with example phrases, and computing dubious, overused patterns. At run-time, as the user types (or mouses over) a word, the system automatically retrieves and displays grammar patterns and examples, most relevant to the word. The user can opt for patterns from a general corpus, academic corpus, learner corpus, or commonly overused dubious patterns found in a learner corpus. *WriteAhead* proactively engages the user with steady, timely, and spot-on information for effective assisted writing. Preliminary experiments show that *WriteAhead* fulfills the design goal of fostering learner independence and encouraging self-editing, and is likely to induce better writing, and improve writing skills in the long run.

1 Introduction

The British Council has estimated that roughly a billion people are learning and using English around the world (British Council 1997), mostly as a second language, and the numbers are growing. Clearly, many of L2 speakers of English feel themselves to be at a disadvantage in work that requires communication in English. For example, Flowerdew (1999) reports that a third of Hong Kong academics feel disadvantged in publishing a paper internationally, as compared to native speakers.

These L2 speakers and learners provide motivation for research and development of computer assisted language learning, in particular tools that help identify and correct learners' writing errors. Much work has been done on developing technologies for automated gramatical error correction (GEC) to assist language learners (Leacock, Chodorow, Gamon, and Tetreault 2010). However, such efforts have not led to the development of a production system (Wampler, 2002).

However, Milton (2010) pointed out that focusing on fully-automatic, high quality GEC solutions has overlooked the long-term pedagogical needs of L2 learner writers. Learners could be more effectively assisted in an interactive writring environment (IWE) that constantly provides context-sensitive writing suggestions, right in the process of writing or self-editing.

Consider an online writer who starts a sentence with *"This paper discusses ...."* The best way the system can help is probably displaying the patterns related to the last word *discuss* such as **discuss something** and **discusses with someone**, that help the user to write accurately and fluently. If the user somehow writes or pastes in some incorrect sentence, *"This paper discusses about the influence of interference and reflection of light."* The best way the system can help is probably displaying the erroneous or overused pattern, **discuss about something**, that prompts the user to change the sentence to *"This paper discusses the influence of interference and reflection of light."*

Intuitively, by extracting and displaying such patterns and examples, distilled from a very large corpus, we can guide the user towards writing flu-

Proceedings of ACL-IJCNLP 2015 System Demonstrations, pages 139–144,
Beijing, China, July 26-31, 2015.

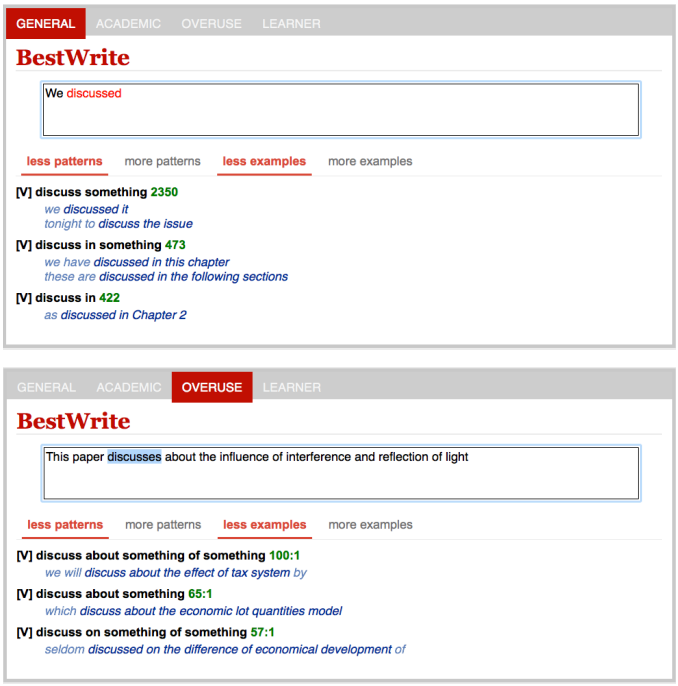

Figure 1: Example *WriteAhead* session where an user typed "This paper present method".

ently, and free of grammatical errors.

We present a new system, *WriteAhead*, that proactively provides just-in-time writing suggestions to assist student writers, while they type away. Example *WriteAhead* suggestions for *"We discussed ..."* are shown in Figure 1. *WriteAhead* has determined the best patterns and examples extracted from the underlying corpus. *WriteAhead* learns these patterns and examples automatically during training by analyzing annotated dictionary examples and automatically tagged sentences in a corpus. As will be described in Section 4, we used the information on collocation and syntax (ICS) for example sentences from online *Macmillan English Dictionary*, as well as in the *Citeseer x* corpus, to develop *WriteAhead*.

At run-time, *WriteAhead* activates itself as the user types in yet another word (e.g., "*discussed*" in the prefix "*We discussed ...*"). *WriteAhead* then retrieves patterns related to the last word. *WriteAhead* goes one step further and re-ranks the suggestions, in an attempt to move most relevant suggestions to the top. *WriteAhead* can be accessed at `http://writehead.nlpweb.org/`.

In our prototype, *WriteAhead* returns the suggestions to the user directly (see Figure 1); alternatively, the suggestions returned by *WriteAhead* can be used as input to an automatic grammar checker or an essay rater.

The rest of this paper is organized as follows. We review the related work in the next section. Then we present our method for automatically learning normal and overused grammar patterns and examples for use in an interactive writing environment (Section 3). Section 5 gives a demonstration script of the interactive writing environment.

2 Related Work

Much work described in a recent survey (Leacock, Chodorow, Gamon, and Tetreault 2010) show that the elusive goal of fully-automatic and high-quality grammatical error correction is far from a reality. Moreover, Milton (2010) pointed out that we should shift the focus and responsibility to the learner, since no conclusive evidence shows explicit correction by a teacher or machine is leads to improved writing skills (Truscott, 1996; Ferris and Hedgcock, 2005). In this paper, we develop an interactive writing environment (IWE) that constantly provides context-sensitive writing suggestions, right in the process of writing or self-editing.

Autocompletion has been widely used in many language production tasks (e.g., search query and translation). Examples include *Google Suggest* and *TransType*, which pioneered the interactive user interface for statistical machine translation (Langlais et al., 2002; Casacuberta et al. 2009). However, previous work focuses exclusively on

Procedure ExtractPatterns(*Sent*, *Keywords*, *Corpus*)

(1) Learning phrase templates for grammar patterns of content words (Section 3.1.1)

(2) Extracting grammar patterns for all keywords in the given corpus based on phrase templates (Section 3.1.2)

(3) Extracting exemplary instances for all patterns of all keywords (Section 3.1.3)

Figure 2: Outline of the pattern extraction process

providing surface suggestions lacking in generality to be truely effective for all users in different writing situation. In contrast, we provide suggestions in the form of theoretical and pedagogically sound language representation, in the form of Pattern Grammar (Hunston and Francis 2000). We also provide concise examples much like concordance advocated by Sinclair (1991).

Much work has been done in deriving context-free grammar from a corpus, while very little work has been done in deriving pattern grammar. Mason and Hunston (2004) reports on a pilot study to automatically recognize grammar patterns for verbs, using only limited linguistic knowledge. It is unclear whether their method can scale up and extend to other parts of speech. In contrast, we show it is feasible to extract grammar patterns for nouns, verbs, and adjectives on a large scale using a corpus with hundreds of million words.

Researchers have been extracting error patterns in the form of word or part of speech (POS) sequencesto detect real-word spelling errors (e.g., Golding and Schabes, 1996; Verberne, 2002). For example, the sequence of **det. det. n.** definitely indicate an error, while **v. prep. adv.** might or might not indicate an error. For this reason, function words (e.g., prepositions) are not necessarily reduced to POS tags (e.g., **v. to adv.**). Sometimes, even lexicalized patterns are necessary (e.g., **go to adv.**) Sun et al. (2007) extend n-grams to non-continuous sequential patterns allowing arbitrary gaps between words. In a recent study closer to our work, Gamon (2011) use high-order part-of-speech ngram to model and detect learner errors on the sentence level.

In contrast to the previous research in developing computer assisted writing environment, we present a system that automatically learns grammar patterns and examples from an academic written corpus as well as learner corpus, with the goal of providing relevant, in-context suggestions.

3 Method

Non-native speakers often make grammatically error, particularly in using some common words in writing (e.g., *discuss* vs. *discuss *about*). In addition, using dictionaries or mere lexical suggestions to assist learner in writing is often not sufficient, and the information could be irrelevant at times. In this section, we address such a problem. Given various corpora (e.g., *BNC* or *CiteseerX*) in a specific genre/domain and a unfinished or completed sentence, we intend to assist the user by retrieving and displaying a set of suggestions extracted from each corpus. For this, by a simple and intuitional method, we extract grammatical error patterns and correction such that the top ranked suggestions are likely to contain a pattern that fits well with the context of the unfinished sentence. We describe the stage of our solution to this problem in the subsections that followed.

3.1 Extracting Grammar Patterns

We attempt to extract grammatical error patterns and correction for keywords in a given corpus to provide writing suggestions, in order to assist ESL learners in an online writing session. Our extraction process is shown in Figure 2.

3.1.1 *Learning Extraction Templates* In the first stage of the extraction process (Step (1) in Figure 2), we generate a set of phrase templates for identifying grammar patterns based on information on Collocation and Syntax (ICS) in an online dictionary.

For example, the dictionary entry of *difficulty* may provide examples with ICS pattern, such as **have difficulty/problem (in) doing something**: *Six months after the accident, he still has difficulty walking*. This complicated pattern with parenthetical and alternative parts can be expanded to yield patterns such as **have difficulty in doing something**. By generalizing such a pattern into templates with PoS and phrase tags (e.g., **v. np prep. v np**, we can identify instances of such a pattern in tagged and chunked sentences. For this, we expand the parentheticals (e.g., **(in)**) and alternatives (e.g., **difficulty/problem**) in ICS.

Then, we replace (non-entry) words in ICS with the most frequent part of speech tags or phrase tags, resulting in sequences of POS and phrase labels (e.g., **v. difficulty prep. v. np**). Then, we take only the complementation part (e.g., **prep. v. np**). Finally, we convert each complementa-

tion into a regular expression for a *RegExp* chunk parser.

Subsequently, we convert each template into a regular expression of chunk labels, intended to match instances of potential patterns in tagged sentences. The chunk labels typically are represented using B-I-O symbols followed by phrase type, with each symbol denoting **B**eginning, **I**nside, and **O**utside of the phrase. Note that in order to identify the head of a phrase, we change the B-I-O representation to I-H-O, with H signifying the **H**ead.

3.1.2 *Extracting Patterns* In the second stage of the extraction process (Step (2) in Figure 2), we identify instances of potential patterns for all keywords. These instances are generated for each tagged and chunked sentence in the given corpus and for each chunk templates obtained in the previous stage.

We adopt the *MapReduce* framework to extract salient patterns. At the start of the *Map Procedure*, we perform part of speech, lemmatization, and base phrase tagging on the sentences. We then find all pattern instances anchoring at a keyword and matching templates obtained in the first stage. Then, from each matched instance, we extract the tuple, (*grammar pattern*, *collocation*, and *ngrams*). Finally, we emit all tuples extracted from the tagged sentence. The map procedure is applied to every tagged sentence in the given corpus.

In the *reduce* part, the *ReducePattern Procedure* receives a batch of tuples, locally sorted and grouped by keyword, as is usually done in the *MapReduce* paradigm. At the start of the *ReducePattern Procedure*, we further group the tuple by pattern. Then we count the number of tuples of each pattern as well as within-group average and standard deviation of the counts. Finally, with these statistics, we filter and identify patterns more frequent than average by 1 standard deviation. The *ReducePattern Procedure* is applied to all tuples generated in the *Map Procedure*. Sample output of this stage is shown in Table 1.

3.1.3 *Extracting Exemplary Phrases* In the third and final stage of extraction, we generate exemplary phrases for all patterns of all keywords of interest using the *ReduceCollExm Procedure*, which is done after the Map procedure, and essentially the same as the *ReducePattern Procedure* in the second stage (Section 3.1.2).

In the spirit of the GDEX method (Kilgarriff et al. 2008) of selecting good dictionary examples for a headword via collocations, we propose a method for selection good example for a pattern. For this, we count and select salient collocations (e.g., the heads of phrases, *difficulty in process* in pattern instance *difficulty in the optimization process*). For each selected collocation, we choose the most frequent instance (augmented with context) to show the user the typical situation of using the collocation.

Table 1: Example *difficulty* patterns extracted.

Pattern	Count	Example
difficulty of something	2169	*of the problem*
difficulty in doing something	1790	*in solving the problems*
difficulty of doing something	1264	*of solving this problem*
difficulty in something	1219	*in the previous analyses*
difficulty with something	755	*with this approach*
difficulty doing something	718	*using it*

Note: There are 11200 instances of potential *difficulty* patterns with average count of 215 and a standard deviation of 318

These examples also facilitate the system in ranking patterns (as will be described in Section 3.2). For that, we add one chunk before, and one chunk after the collocational instance. For example, the collocation, **method for solution of equation** is exemplified using the authentic corpus example, "*method for exact solution of homogeneous linear differential equation*" in the context of "*report a new analytical ... with*." We use a similar procedure as describe in Section 3.1.2 to extract examples.

After the grammar patterns are extracted from a reference corpus and a learner corpus, we normalize and compared the counts of the same pattern in the two corpora and compuate an overuse ratio for all patterns and retain patterns with a high overuse ratio.

3.2 Retrieving and Ranking Suggestions

Once the patterns and examples are automatically extracted for each keyword in the given corpus, they are stored and indexed by keyword. At runtime in a writing session, *WriteAway* constantly probes and gets the last keyword of the unfinished sentence *Sent* in the text box (or the word under the mouse when in editing mode). With the keyword as a query, *WriteAway* retrieves and ranks all relevant patterns and examples (*Pat* and *Exm*) aiming to move the most relevant information toward the top. We compute the longest common subsequence (LCS) of *Sent* and an example, *Exm*. The examples and patterns are ranked by

Score(*Exm*) = | *LCS*(*Exm*, *Sent*) | × Count(*Exm*).

Score(*Pat*) = $\sum$ Score(*E*), where *E* is an example of *Pat*

To improve ranking, we also try to find the longest similar subsequence (LSS) between the user input, *Sent* and retrieved example, *Exm* based on distributional word similarity using the word2vec (Mikolov et al., 2013) cosine distance. The new score function is:

Score(*Exm*) = *LSS*(*Exm*, *Sent*) × Count(*Exm*),

$LSS(Exm, Sent) = \max sim(Exm_{sub}, Sent_{sub})$,

$sim(A, B) = 0$, if $|A| \neq |B|$.

$sim(A, B) = \sum word\text{-}sim(A_i, B_i)$, otherwise.

4 Experiments and Results

For training, we used a collection of approximately 3,000 examples for 700 headwords obtained from online Macmillan English Dictionary (Rundel 2007), to develop the templates of patterns. The headwords include nouns, verbs, adjectives, and adverbs. We then proceeded to extract writing grammar patterns and examples from the British National Corpus (BNC, with 100 million words), *CiteseerX* corpus (with 460 million words) and Taiwan Degree Thesis Corpus (with 10 million words). First, we used Tsujii POS Tagger (Tsuruoka and Tsujii 2005) to generate tagged sentences. We applied the proposed method to generate suggestions for each of the 700 content keywords in Academic Keyword List.

4.1 Technical Architecture

WriteAhead was implemented in Python and Flask Web framework. We stored the suggestions in JSON format using PostgreSQL for faster access. *WriteAhead* server obtains client input from a popular browser (Safari, Chrome, or Firefox) dynamically with AJAX techniques. For uninterrupted service and ease of scaling up, we chose to host *WriteAhead* on Heroku, a cloud-platform-as-a-service (PaaS) site.

4.2 Evaluating *WriteAhead*

To evaluate the performance of *WriteAhead*, we randomly sampled 100 sentences from a learner corpus with complementation errors. For each sentence, we identify the keyword related to the error and checked whether we have identify an overused pattern relevant to the error, and if positive the rank of this pattern. We then use the Mean Reciprocate Rank (MRR) to measure performance. Evaluation of *WriteAhead* showed a MMR rate of .30 and a recall rate of 24%. The Top 1, 2, 3 recall rates are 31%, 35%, and 38% respectively

5 Demo script

In this demo, we will present a new writing assistance system, *WriteAhead*, which makes it easy to obtain writing tips as you type away. *WriteAhead* does two things really well.

First, it examines the unfinished sentence you just typed in and then automatically gives you tips in the form of grammar patterns (accompanied with examples similar to those found in a good dictionary) for continuing your sentence.

Second, WriteAhead automatically ranks suggestions relevant to your writing, so you spend less time looking at tips, and focus more on writing your text.

You might type in *The paper present method* and you are not sure about how to continue. You will instantly receive tips on grammar as well as content as shown in Figure 1. At a quick glance, you might find a relevant pattern, *method for doing something* with examples such as *This paper presents/describes a method* for generating solutions. That could tip you off as to change the sentence into *This paper presents a method*, thus getting rid of tense and article errors, and help you continue to write something like *method for extracting information*.

Using *WriteAhead* this way, you could at once speed up writing and avoid making common writing errors. This writing and tip-taking process repeats until you finish writing a sentence. And as you start writing a new, the process starts all over again.

Most autocompletion systems such as Google Suggest and TransType offer word-level suggestions, while *WriteAhead* organizes, summarizes, and ranks suggestions, so you can, at a glance, grasp complex linguistic information and make quick decision. Our philosophy is that it is important to show information from general to specific to reduce the cognitive load, so while minding the form, you can still focus on the content of writing.

WriteAhead makes writing easy and fun, and it also turns writing into a continuous learning process by combining problem solving and information seeking together to create a satisfying user experience. *WriteAhead* can even help you beat Writers Block. *WriteAhead* can be accessed at `http://writeahead.nlpweb.org/`.

6 Conclusion

Many avenues exist for future research and improvement of *WriteAhead*. For example, corpora for different language levels, genres (e.g., emails, news) could be used to make the suggestions more relevant to users with diverse proficiency levels and interests. NLP, IR, and machine learning techniques could be used to provide more relevant ranking, to pin-point grammatical errors, or to generate finer-grained semantic patterns (e.g., **assist someone in something** or **attend activity/institution**) Additionally, an interesting direction is identifying grammar patterns using a CRF sequence labeller.

In summary, in an attempt to assist learner writers, we have proposed a method for providing writing suggestion as a user is typewriting. The method involves extracting, retrieving, and ranking grammar patterns and examples. We have implemented and evaluated the proposed method as applied to a scholarly corpus with promising results.

References

Casacuberta, Francisco, et al. "Human interaction for high-quality machine translation." Communications of the ACM 52.10 (2009): 135-138.

Dagneaux, Estelle, Sharon Denness, and Sylviane Granger. "Computer-aided error analysis." System 26.2 (1998): 163-174.

Flowerdew, John. "Problems in writing for scholarly publication in English: The case of Hong Kong." Journal of Second Language Writing 8.3 (1999): 243-264.

Ferris, Dana, and J. S. Hedgcock. "Teacher response to student writing: Issues in oral and written feedback." Teaching ESL composition: Purpose, process and practice (2005): 184-222.

Graddol, David. "The future of English?: A guide to forecasting the popularity of the English language in the 21st century." (1997).

Granger, Sylviane, and Paul Rayson. Automatic profiling of learner texts." Learner English on computer (1998): 119-131.

Granger, Sylviane, and Stephanie Tyson. "Connector usage in the English essay writing of native and non-native EFL speakers of English." World Englishes 15.1 (1996): 17-27.

Golding, Andrew R., and Yves Schabes. "Combining trigram-based and feature-based methods for context-sensitive spelling correction." Proceedings of the 34th annual meeting on Association for Computational Linguistics. Association for Computational Linguistics, 1996.

Gamon, Michael. "High-order sequence modeling for language learner error detection." Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications. Association for Computational Linguistics, 2011.

Hunston, Susan, and Gill Francis. Pattern grammar: A corpus-driven approach to the lexical grammar of English. Amsterdam: John Benjamins, 2000.

Leacock, Claudia, et al. "Automated grammatical error detection for language learners." Synthesis lectures on human language technologies 3.1 (2010): 1-134.

Milton, John, and Vivying SY Cheng. "A toolkit to assist L2 learners become independent writers." Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics and Writing: Writing Processes and Authoring Aids. Association for Computational Linguistics, 2010.

Mason, Oliver, and Susan Hunston. "The automatic recognition of verb patterns: A feasibility study." International journal of corpus linguistics 9.2 (2004): 253-270.

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in Neural Information Processing Systems. 2013.

Sun, Guihua, et al. "Detecting erroneous sentences using automatically mined sequential patterns." Annual Meeting-Association for Computational Linguistics. Vol. 45. No. 1. 2007.

Truscott, John. "The case against grammar correction in L2 writing classes." Language learning 46.2 (1996): 327-369.

Tsuruoka, Yoshimasa, and Jun'ichi Tsujii. "Chunk parsing revisited." Proceedings of the Ninth International Workshop on Parsing Technology. Association for Computational Linguistics, 2005.

Verberne, Suzan. "Context-sensitive spell checking based on word trigram probabilities." Unpublished masters thesis, University of Nijmegen (2002).

Sinclair J. (1991) Corpus, Concordance, Collocation. Oxford University Press, Hong Kong.

P. Langlais, G. Foster, and G. Lapalme. 2000. TransType: a computer-aided translation typing system. In Workshop on Embedded Machine Translation Systems.

Caragea, Cornelia, et al. "CiteSeer x: A Scholarly Big Dataset." Advances in Information Retrieval. Springer International Publishing, 2014. 311-322.