WriteAhead2: Mining Lexical Grammar Patterns for Assisted Writing

Jim Chang
Department of Computer Science,
National Tsing Hua University
101, Kuangfu Road,
Hsinchu, 300, Taiwan
jim.chang.nthu@gmail.com

Jason S. Chang
Department of Computer Science,
National Tsing Hua University
101, Kuangfu Road,
Hsinchu, 300, Taiwan
jason.jschang@gmail.com

Abstract

This paper describes *WriteAhead2*, an interactive writing environment that provides lexical and grammatical suggestions for second language learners, and helps them write fluently and avoid common writing errors. The method involves learning phrase templates from dictionary examples, and extracting grammar patterns with example phrases from an academic corpus. At run-time, as the user types word after word, the actions trigger a list after list of suggestions. Each successive list contains grammar patterns and examples, most relevant to the half-baked sentence. *WriteAhead2* facilitates steady, timely, and spot-on interactions between learner writers and relevant information for effective assisted writing. Preliminary experiments show that *WriteAhead2* has the potential to induce better writing and improve writing skills.

1 Introduction

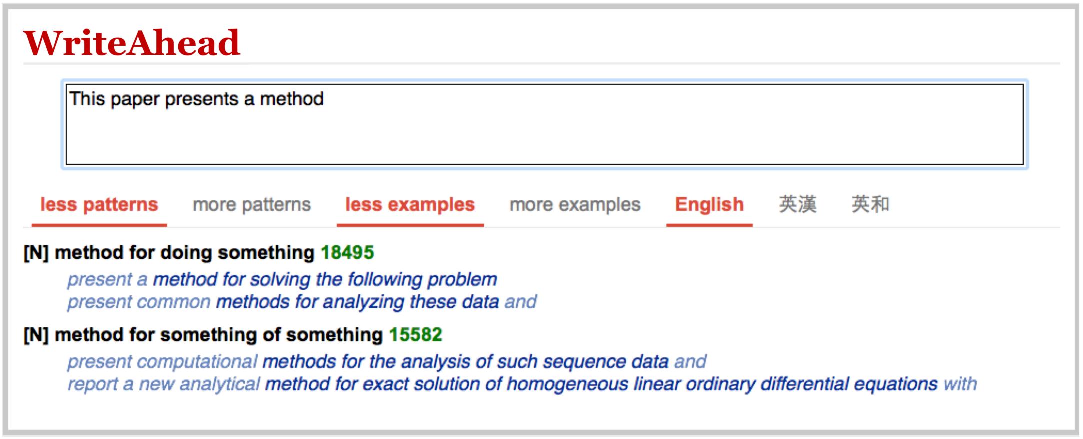
More and more non-native speakers are writing in English as a second language for global communication, especially in academia. Unavoidably, these L2 writers encounter many problems: in form and content, in grammar, style, and discourse. Much work has been done on developing computer-assisted language reference tools to improve L2 learners' writing skills. Furthermore, many researchers have worked on providing corrective feedback and grades, by automatically detecting and correcting grammatical errors in learner writings.

Computer assisted language learning (CALL) systems typically help the users *before* and *after* writing. For example, *NetSpeak* (`www.netspeak.org`) uses Google Web 1T Corpus to retrieve common phrases relevant to a user query, while *Marking Mate* (`www.readingandwritingtools.com`) accepts an user essay and offers a grade with corrective feedback. However, learner writers sorely need concrete writing suggestions, right in the context of writing. Learners could be more effectively assisted, if CALL tools provide such suggestions as learners write away.

Consider an online writer who is composing a sentence starting with "*We propose a method ...*" The best way the system can help is probably not just dictionary-lookup, but rather in-page suggestions tailor-made for this *very* context of continuing the unfinished sentence. Furthermore, fixed-length ngrams such as (1) *method for automatic evaluation* and (2) *method for determining the* is not good enough, or *general enough*, for all writers addressing diverse issues.

Appropriate suggestions should contains *general*, *long* grammar patterns such as: (1) **method for doing something**: *method for finding a solution* (2) **method for something**: *method for grammatical error detection*. Intuitively, by extracting and displaying such patterns and examples, distilled from a very large corpus, we can guide the user towards writing fluently, and free of grammatical errors.

We present a new system, *WriteAhead2*, that proactively provides just-in-time writing suggestions to assist student writers, while they type away. *WriteAhead2* is a continuation of the work of *WriteAhead* (Liou, Yang, Chang 2012). Example *WriteAhead2* suggestions for *"We propose a method*


WriteAhead

This paper presents a method

[N] method for doing something 18495

present a ***method for solving the following problem***

present common ***methods for analyzing these data*** *and*

[N] method for something of something 15582

present computational ***methods for the analysis of such sequence data*** *and*

report a new analytical ***method for exact solution of homogeneous linear ordinary differential equations*** *with*

Figure 1: Example *WriteAhead2* session.

...” are shown in Figure 1. *WriteAhead2* has determined the best patterns and examples extracted from the underlying corpus. *WriteAhead2* 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 *WriteAhead2*.

At run-time, *WriteAhead2* activate itself as the user types in yet another word (e.g., ”*method*” in the prefix ”*We propose a method ...*”). *WriteAhead2* then retrieves patterns related to the last word. *WriteAhead2* goes one step further and re-ranks the suggestions, in an attempt to move most relevant suggestions to the top. *WriteAhead2* can be accessed at `http://writeahead.nlpweb.org`.

In our prototype, *WriteAhead2* returns the suggestions to the user directly (see Figure 1); alternatively, the suggestions returned by *WriteAhead2* can be used as input to an automatic grammar checker or an essay rater.

2 Related Work

Using dictionaries for language learning has a long and honorable history. However, Sinclair (1991) pointed out dictionaries are limited by their narrow focus on meaning, lack in pragmatics, and insufficient genre/discipline-specific information. Therefore, Sinclair advocated corpus linguistics, corpus-based lexicography, and using a concordance in language teaching.

In the area of corpus-based language learning, Weber (2001) illustrated how combining learner writing and a concordance helped law students in writing better legal essays. Sun (2007) proposed a web-based *Scholarly Writing Template* (SWT) system for graduate students based on a small, manually-annotated corpus. In contrast, we focus on grammar, the most problematic area for learners.

In the area of automated essay rating, *Criterion* (Burstein, Chodorow and Leacock, 2003) uses statistical models to evaluate student writing and provides corrective feedback. Criterion has been used for rating 4 to 12th graders’ writings, and TOFEL/GRE composition tests. *Criterion* handles essay writing, while *WriteAhead2* concentrates on helping learner with the genre of research articles.

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, Foster and Lapalme, 2002).

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, with the goal of providing relevant, in-context suggestions.

3 Method

Often, it is not sufficient to use dictionaries or lexical autocompletion to assist learner in writing. Unfortunately, very few Language tools offer compre-

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

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

(2) Extracting 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

hensive writing suggestions during writing. In this section, we address such a problem. Given a corpus in a specific genre/domain (e.g., *Citeseer x*), and an unfinished sentence, we intend to assist the user by retrieving and displaying a set of suggestions based on the corpus. For this, we extract grammar patterns with exemplary instances from the corpus. We describe the stages of our solution to this problem in the subsections that followed.

3.1 Extracting Grammar Patterns

We attempt to extract characteristic grammar patterns for keywords in a given corpus to provide writing suggestions, for L2 learners in an online writing session. The set of keywords (as will be described in Section 4) include the words academic writers use most frequently for rhetoric purposes, including stating a topic, hypothesizing, contrasting, exemplifying, explaining, evaluating and other functions. Our extraction process is shown in Figure 2.

3.1.1 *Learning Templates of Grammar Patterns* In the first stage of the extraction process (Step (1) in Figure 2), we generate a set of phrase templates for identifying grammar patterns. For example, a dictionary example with ICS—**have difficulty/problem (in) doing something**: *Six months after the accident, he still has difficulty walking*, implies that this pattern (i.e. **have difficulty in doing something**) can realize in a phrase sequences, ”VP NP prep. VP NP”. With such a template, we can identify potential patterns for verbs and nouns (e.g., *differ* or *difficulty*). We expand the parentheticals (e.g., **(in)**) and alternatives (e.g., **difficulty/problem**) in ICS, and keep only the most frequent templates. Finally, each of these templates is converted into a regular expression for a *RegExp* chunk parser.

3.1.2 *Extracting Patterns* In the second stage of the extraction process (Step (2) in Figure 2), we identify instances of potential pattern for all keywords. These instance 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 characteristic patterns. In Step (1) of the Map procedure, we perform part of speech 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. Note that matching is done on the sequence of BIO phrase labels (denoting **B**eginning, **I**nside, and **O**utside of base NP, VP, PP, and ADJP). Then from each matched instance, we extract a tuple of keyword, POS, grammar pattern, collocates (of the keyword), and ngram (word sequence) in Steps (4a) through (4c). Finally, we emit all tuples extracted from the tagged sentence (Step (5)).

The Reduce procedure receives a batch of hashed and locally sorted tuples, grouped by the head word and POS. In Step (1) of the Reduce procedure, we further group the tuples by pattern. Then we count the number of tuples of each pattern (in Step (2)) as well as within-group average and standard deviation (in Step (3)). Finally, With these statistics, we filter and identify patterns more frequent than average by K standard deviation, $K = 1$ (in Step (4)), following Smadja (1993).

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. The procedure is essentially the same as the Reduce procedure in the second stage (Section 3.1.2).

3.2 Retrieving and Ranking Suggestions

Once the patterns and examples are automatically extracted for each keyword in the given corpus, they are stored as suggestions for the last word the user types in. *WriteAhead2* constantly probes and gets the last written word from the writing area. With the last word as a query, *WriteAhead2* retrieves patterns and examples, and re-ranks the results to move the most relevant information toward the top.

Currently, we re-rank patterns by using word overlap between the last written sentence and the retrieved examples. When there is no word overlap, we fall back to frequency-based ranking. An exam-

```
Procedure Map(Sent, AKL, Template)
(1)  TaggedSent = TagAndChunkParse(Sent)
     For each Word $\in$ AKL at position i in TaggedSent
(2)      Match = RegExpChunkParse(TaggedSent, Template, i)
         If Match is found
(3)          ChunkedPhr = CutChunk(TaggedSent, i, Match)
(4a)         Pat = ExtractPattern(ChunkedPhr)
(4b)         Col = ExtractCollocation(ChunkedPhr)
(4c)         Ng  = ExtractNgram(ChunkedPhr)
(5)          Emit Tuple = (Word, Pat, Col, Ng)

Procedure Reduce(Tuples for a word)
(1)  Pats, PatTuples = GroupTuplesByPat(Tuple)
(2)  Pats, Counts = Counter(Pats, PatTuples)
(3)  Avg, STD = CalStatatics(Pats, Counts)
     For each Pat, Count pair in (Pats, Counts)
         If $Count > Avg + K \times STD$
(4)          Emit Tuple = (Word, Pat, PatTuples)
```

Fig. 3. Outline of the process used to extract CPs.

ple session is shown in Figure 1.

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 generate writing suggestions from the *Citeseer x* corpus. 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

WriteAhead2 was implemented in Python and Flask Web framework. We stored the suggestions in JSON format using PostgreSQL for faster access. *WriteAhead2* 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 *WriteAhead2* on Heroku, a cloud-platform-as-a-service (PaaS) site.

Table 1: Human evaluation of *WriteAhead2*

Suggestion	Count	Percent	Recall
1st suggestion	141	.53	.43
2nd suggestion	50	.19	.15
3rd suggestion	38	.14	.12
Top 3 suggestions	229	.85	.70
Not in Top 3	38	—	.12
No suggestions	62	—	.19
Not applicable	71	—	—

4.2 Evaluating *WriteAhead2*

To evaluate the performance of *WriteAhead2*, we randomly sampled sentences from conference papers. For simplicity, we tested if our system can provide proper grammar patterns for the first noun or verb in theses sentence. We randomly selected 400 sentences from ACL-2014 long papers. For each sentence, we pasted the sentence prefix up to the the first (noun or verb) keyword to the input box of *WriteAhead2*. The reason for targeting verbs and nouns is that they are considered as exhibiting regularity in local syntax (Hunston and Francis 2000) and common source of learners' writing errors (De Cock, Gilquin, Granger, Lefer, Paquot, and Ricketts 2007). Finally, we manually determined the appropriateness of suggestions for continuing part of the sentence based on the precision of the Top-3 suggestions. For example, we took a sentence:

> *There is some prior work on the related task of hierarchical clustering, or grouping together of semantically related words ...*

and identified the first noun or verb (e.g., *work*) as the anchor, and run *WriteAhead2* on the prefix ending at the anchor (e.g, "*There is some prior work*"). The Top 3 suggestions displayed by *WriteAhead2* were than examined by a human judge to evaluate for correctness in predicting what follow the prefix. For instance, if the first suggestion is:

> **work on something of something 1332:** *VoiSe is designed to work on a symbolic representation of a music score*

Then the judge would determine it is a correct prediction of *work on the related task of hierarchical clustering* and record that the first suggestion is correct. Evaluation of *WriteAhead2* showed a Top 1 precision rate of 53% and recall rate of 70% when considering the Top 3 suggestions.

5 Demo Script

In this demo, we will present a new writing assistance system, *WriteAhead2*, which makes it easy to obtain writing tips as you type away. *WriteAhead2* 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, *WriteAhead2* automatically ranks suggestions relevant to your writing, so you spend less time looking at tips, and focus more on writing your piece.

You might type in *This paper presents a method* and 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 *WriteAhead2* 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 *WriteAhead2* 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.

6 Conclusion

Many avenues exist for future research and improvement of *WriteAhead2*. 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. Yet another direction of research would be to extract and display backward-looking suggestions to complement the current forward-looking suggestions.

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

Jill Burstein, Martin Chodorow, and Claudia Leacock. 2003. *Criterion: Online Essay Evaluation–An Application for Automated Evaluation of Student Essays.* In Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence. Acapulco, Mexico.

Cornelia Caragea, Jian Wu, Alina Ciobanu, Kyle Williams, Juan Fernndez-Ramrez, Hung-Hsuan Chen, Zhaohui Wu, and Lee Giles. *CiteSeer x: A Scholarly Big Dataset.* Advances in Information Retrieval. Springer International Publishing, 2014. 311-322.

Sylvie De Cock, Gatanelle Gilquin, Sylviane Granger, Marie-Aude Lefer, Magali Paquot, and Suzanne Ricketts. 2007. *Improve Your Writing Skills.* In Rundell (2007).

Philippe Langlais, George Foster, and Guy Lapalme. 2000. *TransType: A Computer-Aided Translation Typing System.* In Workshop on Embedded Machine Translation Systems.

Hien-Chin Liou, PingChe. Yang, and Jason S. Chang. *Language supports for journal abstract writing across disciplines.* Journal of Computer Assisted Learning 28.4 (2012): 322-335.

Michael Rundell (Ed.). 2007. *Macmillan English Dictionary for Advanced Learners.* Oxford, Macmillan, 2002.

John Sinclair. 1991. *Corpus, Concordance, Collocation.* Oxford University Press, Hong Kong.

Yu-Chih Sun. 2007. *Learner Perceptions of a Concordancing Tool for Academic Writing.* Computer Assisted Language Learning 20, 323343.

Jean-Jacques Weber. 2001. *A Concordance- and Genre-informed Approach to ESP Essay Editing.* ELT Journal 55, 1420.