GRASP: Grammar- and Syntax-based Pattern-Finder in CALL

Chung-Chi Huang[*] **Mei-Hua Chen**[*] **Shih-Ting Huang**[+] **Hsien-Chin Liou**[**] **Jason S. Chang**[+]
[*] Institute of Information Systems and Applications, NTHU, HsinChu, Taiwan, R.O.C. 300
[+] Department of Computer Science, NTHU, HsinChu, Taiwan, R.O.C. 300
[**] Department of Foreign Languages and Literature, NTHU, HsinChu, Taiwan, R.O.C. 300
{u901571,chen.meihua,koromiko1104,hsienchin,jason.jschang}gmail.com

Abstract

We introduce a method for learning to describe the attendant contexts of a given query for language learning. In our approach, we display phraseological information in the form of a summary of *general* patterns as well as lexical bundles anchored at the query. The method involves syntactical analyses and inverted file construction. At run-time, grammatical constructions and their lexical instantiations characterizing the usage of the given query are generated and displayed, aimed at improving learners' deep vocabulary knowledge. We present a prototype system, GRASP, that applies the proposed method for enhanced collocation learning. Preliminary experiments show that language learners benefit more from GRASP than conventional dictionary lookup. In addition, the information produced by GRASP is potentially useful information for automatic or manual editing process.

1 Introduction

Many learners submit word or phrase queries (e.g., "*role*") to language learning sites on the Web to get usage information every day, and an increasing number of services on the Web specifically target such queries. Language learning tools such as concordancers typically accept single-word queries and respond with example sentences containing the words. There are also collocation reference tools such as Sketch Engine and TANGO that provide co-occurring words for the query word. Another collocation tool, JustTheWord further organizes and displays collocation clusters.

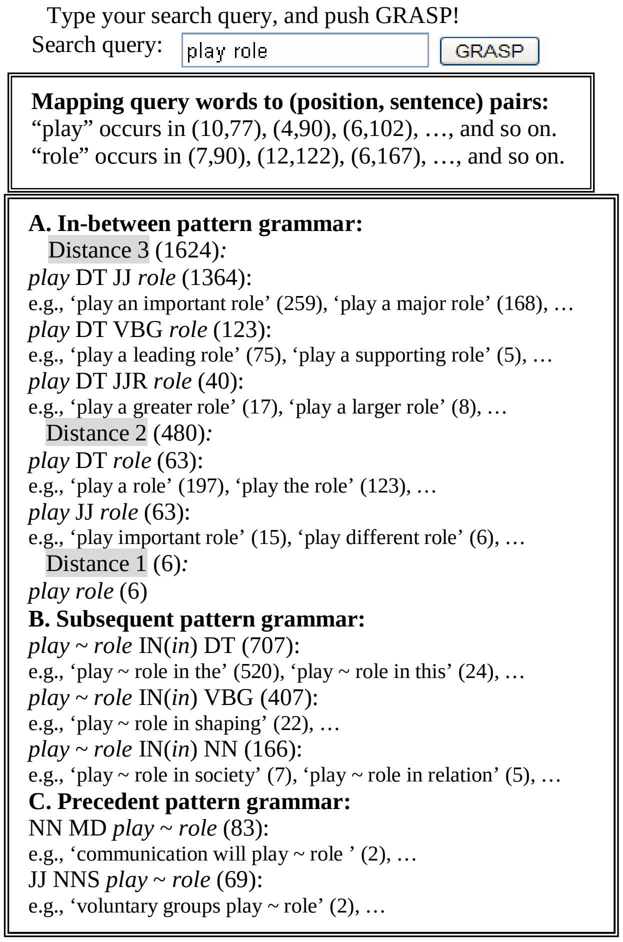
Learners may want to submit phrase queries (fixed or rigid collocaions) to learn further how to use the phrase in context, or in other words, to acquire the knowledge on the attendant phraseology of the query. These queries could be answered more appropriately if the tool accepted long queries and returned a concise summary of their surrounding contexts.

Consider the query "*play role*". The best responses for this query are probably not just example sentences, but rather the phraseological tendencies described grammatically or lexically. A good response of such a summary might contain patterns such as "*play* Det Adj *role*" (as in "*play an important role*") and "*play ~ role in* V-ing" (as in "*play ~ role in shaping …*"). Intuitively, by exploiting simple part-of-speech analysis, we can derive such patterns, inspired by the grammatical theory of Pattern Grammar[1] in order to provide more information on demand beyond what is given in a grammar book.

We present a system, GRASP, that provide a usage summary of the contexts of the query in the form of patterns and frequent lexical bundles. Such rich information is expected to help learners and lexicographers grasp the essence of word usages. An example GRASP response for the query "*play*

[1] Please refer to (Hunston and Francis, 2000).

role” is shown in Figure 1. GRASP has retrieved the sentences containing the query in a reference corpus. GRASP constructs these query-to-sentence index in the preparation stage (Section 3).


Type your search query, and push GRASP!

Search query: play role GRASP

Mapping query words to (position, sentence) pairs:
“play” occurs in (10,77), (4,90), (6,102), …, and so on.
“role” occurs in (7,90), (12,122), (6,167), …, and so on.

A. In-between pattern grammar:
Distance 3 (1624)*:*
play DT JJ *role* (1364):
e.g., ‘play an important role’ (259), ‘play a major role’ (168), …
play DT VBG *role* (123):
e.g., ‘play a leading role’ (75), ‘play a supporting role’ (5), …
play DT JJR *role* (40):
e.g., ‘play a greater role’ (17), ‘play a larger role’ (8), …
Distance 2 (480)*:*
play DT *role* (63):
e.g., ‘play a role’ (197), ‘play the role’ (123), …
play JJ *role* (63):
e.g., ‘play important role’ (15), ‘play different role’ (6), …
Distance 1 (6)*:*
play role (6)
B. Subsequent pattern grammar:
play ~ role IN(*in*) DT (707):
e.g., ‘play ~ role in the’ (520), ‘play ~ role in this’ (24), …
play ~ role IN(*in*) VBG (407):
e.g., ‘play ~ role in shaping’ (22), …
play ~ role IN(*in*) NN (166):
e.g., ‘play ~ role in society’ (7), ‘play ~ role in relation’ (5), …
C. Precedent pattern grammar:
NN MD *play ~ role* (83):
e.g., ‘communication will play ~ role ’ (2), …
JJ NNS *play ~ role* (69):
e.g., ‘voluntary groups play ~ role’ (2), …

Figure 1. An example GRASP search for “play role”.

At run-time, GRASP starts with a search query (e.g., “*play role*”) submitted by the user. GRASP then retrieves example sentences and generates a summary of representative contexts, using patterns (e.g., “*play ~ role in* V-ing”) and lexical bundles (e.g., “*play ~ role in shaping*. In our implementation, GRASP also returns the translations and the example sentences of the lexical instances, so the learner can use their knowledge of native language to enhance the learning process.

2 Related Work

Computer-assisted language learning (CALL) has been an area of active research. Recently, more and more research based on natural language processing techniques has been done to help language learners. In our work, we introduce a language learning environment, where summarized usage information are provided, including how function words and verb forms are used in combination with the query. These usage notes often help contrast the common sources of error in learners’ writing (Nicholls, 1999). In our pilot teaching experiment, we found learners have problems using articles and prepositions correctly in sentence composition (as high as 80% of the articles and 60% of the prepositions were used incorrectly), and GRASP is exactly aimed at helping ESL or EFL learners in that area.

Until recently, collocations and usage information are compiled mostly manually (Benson et al., 1986). With the accessibility to large-scale corpora and powerful computers, it has become common place to compile a list of collocations automatically (Smadja, 1993). In addition, there are many collocation checkers developed to help non-native language learners (Chang et al., 2008), or learners of English for academic purposes (Durrant, 2009).

Recently, automatic generation of collocations for computational lexicography and online language learning has drawn much attention. Sketch Engine (Kilgarriff et al., 2004) summarizes a word’s grammatical and collocation behavior, while JustTheWord clusters the co-occurring words of single-word queries and TANGO (Jian et al., 2004) accommodates cross-lingual collocation searches. Moreover, Cheng et al. (2006) describe how to retrieve mutually expected words using concgrams. In contrast, GRASP, going one step further, automatically computes and displays the information that reveals the regularities of the contexts of user queries in terms of grammar patterns.

Recent work has been done on incorporating word class information into the analyses of phraseological tendencies. Stubbs (2004) introduces phrase-frames, which are based on lexical ngrams with variable slots, while Wible et al. (2010) describe a database called StringNet, with lexico-syntactic patterns. Their methods of using word class information are similar in spirit to our work. The main differences are that our patterns is anchored with query words directly and generalizes query’s contexts via parts-of-speech, and that we present the query’s usage summary in terms of function words as well as content word form (e.g., "*play ~ role in* V-ing"), as well as elastic lexical bundles (e.g., "*play ~ role in shaping*"). Additionally, we also use semantic codes (e.g., PERSON) to provide more information in a way similar what is provided in learner dictionaries.

3 The GRASP System

3.1 Problem Statement

We focus on constructing a usage summary likely to explain the contexts of a given linguistic search. The usage summary, consisting of the query's predominant attendant phraseology ranging from pattern grammar to lexical phrases, is then returned as the output of the system. The returned summary, or a set of patterns pivoted with both content and function words, can be used for learners' benefits directly, or passed on to an error detection and correction system (e.g., (Tsao and Wible, 2009) and some modules in (Gamon et al., 2009) as rules. Therefore, our goal is to return a reasonable-sized set of lexical and grammatical patterns characterizing the contexts of the query. We now formally state the problem that we are addressing.

Problem Statement: We are given a reference corpus *C* from a wide range of sources and a learner search query *Q*. Our goal is to construct a summary of word usages based on *C* that is likely to represent the lexical or grammatical preferences on *Q*'s contexts. For this, we transform the words in *Q* into sets of (word position, sentence record) pairs such that the context information, whether lexically- or grammatical-oriented, of the querying words is likely to be acquired efficiently.

In the rest of this section, we describe our solution to this problem. First, we define a strategy for preprocessing our reference corpus (Section 3.2). Then, we show how GRASP generates contextual patterns, comprising the usage summary, at run-time (Section 3.3).

3.2 Corpus Preprocessing

We attempt to find the word-to-sentence mappings and the syntactic counterparts of the L1 sentences expected to speed up run-time pattern generation. Our preprocessing procedure has two stages.

Lemmatizing and PoS Tagging. In the first stage, we lemmatize each sentence in the reference corpus *C* and generate its most probable POS tag sequence. The goal of lemmatization is to reduce the impact of morphology on statistical analyses while that of POS tagging is to provide a way to grammatically describe and generalize the contexts/usages of a linguistic query. Actually, using POS tags is quite natural: they are often used for general description in grammar books, such as *one's* (i.e., possessive pronoun) in the phrase "make up one's mind", *oneself* (i.e., reflexive pronoun) in "enjoy oneself very much", *superlative_adjective* in "the most superlative_adjective", *NN* (i.e., noun) and *VB* (i.e., base form of a verb) in "insist/suggest/demand that NN VB" and so on.

Constructing Inverted Files. In the second stage, we build up inverted files of the lemmas in *C* for quick run-time search. For each lemma, we record the sentences and positions in which it occurs. Additionally, its corresponding surface word and POS tag are kept for run-time pattern grammar generation.

```
procedure GRASPusageSummaryBuilding(query,proximity,N,C)
(1)  queries=queryReformulation(query)
(2)  GRASPresponses= $\phi$
     for each query in queries
(3)    interInvList=findInvertedFile($w_1$ in query)
       for each lemma $w_i$ in query except for $w_1$
(4)      InvList=findInvertedFile($w_i$)
         //AND operation on interInvList and InvList
(5a)     newInterInvList= $\phi$ ; i=1; j=1
(5b)     while i<=length(interInvList) and j<=lengh(InvList)
(5c)       if interInvList[i].SentNo==InvList[j].SentNo
(5d)         if withinProximity(interInvList[i].
                        wordPosi,InvList[j].wordPosi,proximity)
(5e)           Insert(newInterInvList, interInvList[i],InvList[j])
             else if interInvList[i].wordPosi<InvList[j].wordPosi
(5f)           i++
             else //interInvList[i].wordPosi>InvList[j].wordPosi
(5g)           j++
           else if interInvList[i].SentNo<InvList[j].SentNo
(5h)         i++
           else //interInvList[i].SentNo>InvList[j].SentNo
(5i)         j++
(5j)     interInvList=newInterInvList
       //construction of GRASP usage summary for this query
(6)    Usage= $\phi$
       for each element in interInvList
(7)      Usage+={PatternGrammarGeneration(query,element,C)}
(8a)   Sort patterns and their instances in Usage in descending order
                                                    of frequency
(8b)   GRASPresponse=the N patterns and instances in Usage with
                                                 highest frequency
(9)    append GRASPresponse to GRASPresponses
(10) return GRASPresponses
```

Figure 2. Generating pattern grammar and usage summary at run-time.

3.3 Run-Time Usage Summary Construction

Once the word-to-sentence mappings and syntactic analyses are obtained, GRASP generates the usage summary of a query using the procedure in Figure 2.

In Step (1) we reformulate the user query into new ones, *queries*, if necessary. The first type of query reformulation concerns the language used in *query*. If it is not in the same language as *C*, we translate *query* and append the translations to *queries* as if they were submitted by the user. The second concerns the length of the query. Since single words may be ambiguous in senses and contexts or grammar patterns are closely associated with words' meanings (Hunston and Francis, 2000), we transform single-word queries into their collocations, particularly focusing on one word sense (Yarowsky, 1995), as stepping stones to GRASP patterns. Notice that, in implementation, users may be allowed to choose their own interested translation or collocation of the *query* for usage learning. The prototypes for first-language (i.e., Chinese) queries and English queries of any length are at A[2] and B[3] respectively. The goal of cross-lingual GRASP is to assist EFL users even when they do not know the words of their searches and to avoid incorrect queries largely because of miscollocation, misapplication, and misgeneralization.

Afterwards, we initialize *GRASPresponses* to collect usage summaries for *queries* (Step (2)) and leverage inverted files to extract and generate each *query*'s syntax-based contexts. In Step (3) we prep *interInvList* for the intersected inverted files of the lemmas in *query*. For each lemma w_i within, we first obtain its inverted file, *InvList* (Step (4)) and perform an AND operation on *interInvList* (intersected results from previous iteration) and *InvList* (Step (5a) to (5j)[4]), defined as follows.

First, we enumerate the inverted lists (Step (5b)) after the initialization of their indices *i* and *j* and temporary resulting intersection *newInterInvList* (Step (5a)). Second, we incorporate a new instance of (position, sentence), based on *interInvList*[*i*] and *InvList*[*j*], into *newInterInvList* (Step (5e)) if the sentence records of the indexed list elements are the same (Step (5c)) and the distance between their words are within *proximity* (Step (5d)). Otherwise, *i* and *j* are moved accordingly. To accommodate the contexts of queries' positional variants (e.g., "*role to play*" and "*role ~ play by*" for the query "play role"), Step (5d) considers the *absolute* distance. Finally, *interInvList* is set for the next AND iteration (Step (5j)).

Once we obtain the sentences containing *query*, we construct its context summary as below. For each *element*, taking the form ([wordPosi(w_1), …, wordPosi(w_n)], *sentence record*) denoting the positions of *query*'s lemmas in the *sentence*, we generate pattern grammar involving replacing words in the sentence with POS tags and words in wordPosi(w_i) with lemmas, and extracting fixed-window[5] segments surrounding *query* from the transformed sentence. The result is a set of grammatical patterns with counts. Their lexical realizations also retrieved and displayed.

The procedure finally generates top *N* predominant syntactic patterns and their *N* most frequent lexical phrases as output (Step (8)). The usage summaries GRASP returns are aimed to accelerate EFL learners' language understanding and learning and lexicographers' word usage navigation. To acquire more semantic-oriented patterns, we further exploit WordNet and majority voting to categorize words, deriving the patterns like "*provide* **PERSON** *with*."

4 Experimental Results

GRASP was designed to generate usage summarization of a query for language learning. As such, GRASP will be evaluated over CALL. In this section, we first present the setting of GRASP (Section 4.1) and report the results of different consulting systems on language learning in Section 4.2.

4.1 Experimental Setting

We used British National Corpus (BNC) as our underlying reference corpus *C*. It is a British English text collection. We exploited GENIA tagger to obtain the lemmas and POS tags of *C*'s sentences. After lemmatizing and syntactic analyses, all sentences in BNC were used to build up inverted files and used as examples for grammar pattern extraction.

[2] http://140.114.214.80/theSite/bGRASP_v552/

[3] http://140.114.214.80/theSite/GRASP_v552/

[4] These steps only hold for sorted inverted files.

[5] Inspired by (Gamon and Leacock, 2010).

English (E) sentence with corresponding Chinese (C) translation	answer to 1st blank	answer to 2nd blank
C: 環境保護對地球有深遠的影響 E: Environmental protection has ___ impact ___.	a profound	on the Earth
C: 房屋仲介商在賣屋上大賺一筆 E: The real estate agent ___ record profit ___.	made a	on house selling
C: 他們打算在不久將來推出新專輯 E: They plan to release their new album in ___ future	the near	none
C: 他為了再見她一面等了很久 E: He waited for her for a long time in ___ attempt ___ again.	an	to see her

4.2 Results of Constrained Experiments

In our experiments, we showed GRASP[6] to two classes of Chinese EFL (first-year) college students. 32 and 86 students participated, and were trained to use GRASP and instructed to perform a sentence translation/composition task, made up of pretest and posttest. In (30-minute) pretest, participants were to complete 15 English sentences with Chinese translations as hints, while, in (20-minute) posttest, after spending 20 minutes familiarizing word usages of the test candidates from us by consulting traditional tools or GRASP, participants were also asked to complete the same English sentences. We refer to the experiments as constrained ones since the test items in pre- and post-test are the same except for their order. A more sophisticated testing environment, however, are to be designed.

Each test item contains one to two blanks as shown in the above table. In the table, the first item is supposed to test learners' knowledge on the adjective and prepositional collocate of "*have impact*" while the second test the verb collocate *make*, subsequent preposition *on*, and preceding article *a* of "*record profit*". On the other hand, the third tests the ability to produce the adjective enrichment of "in future", and the fourth the in-between article *a* or possessive *his* and the following infinitive of "*in attempt*". Note that as existing collocation reference tools retrieve and display collocates, they typically ignore function words like articles and determiners, which happen to be closely related to frequent errors made by the learners (Nicholls, 1999), and fail to provide an overall picture of word usages. In contrast, GRASP attempts to show the overall picture with appropriate function words and word forms.

We selected 20 collocations and phrases [7] manually from 100 most frequent collocations in BNC whose MI values exceed 2.2 and used them as the target for learning between the pretest and posttest. To evaluate GRASP, half of the participants were instructed to use GRASP for learning and the other half used traditional tools such as online dictionaries or machine translation systems (i.e., Google Translate and Yahoo! Babel Fish). We summarize the performance of our participants on pre- and post-test in Table 1 where *GRASP* denotes the experimental group and *TRAD* the control group.

	class 1		*class 2*		*combined*	
	pretest	posttest	pretest	posttest	pretest	posttest
GRASP	26.4	**41.9**	43.6	**58.4**	38.9	**53.9**
TRAD	27.1	32.7	43.8	53.4	39.9	48.6

Table 1. The performance (%) on pre- and post-test.

We observe in Table 1 that (1) the partition of the classes was quite random (the difference between *GRASP* and *TRAD* was insignificant under pretest); (2) GRASP summaries of words' contexts were more helpful in language learning (across *class 1*, *class 2* and *combined*). Specifically, under the column of the 1st class, GRASP helped to boost students' achievements by 15.5%, almost *tripled* (15.5 vs. 5.6) compared to the gain using *TRAD*; (3) the effectiveness of GRASP in language learning do not confine to students at a certain level. Encouragingly, *both* high- and low-achieving students benefited from GRASP if we think of students in *class 2* and those in *class 1* as the high and the low respectively (due to the performance difference on pretests).

We have analyzed some participants' answers and found that GRASP helped to reduce learners' article and preposition errors by 28% and 8%, comparing to much smaller error reduction rate 7% and 2% observed in *TRAD* group. Additionally, an experiment where Chinese EFL students were asked to perform the same task but using GRASP as well as GRASP with translation information[8]

[6] http://koromiko.cs.nthu.edu.tw/grasp/
[7] Include the 15 test items.

[8] http://koromiko.cs.nthu.edu.tw/grasp/ch

was conducted. We observed that with Chinese translation there was an additional 5% increase in students' test performance. This suggests to some extent learners still depend on their first languages in learning and first-language information may serve as another quick navigation index even when English GRASP is presented.

Overall, we are modest to say that (in the constrained experiments) GRASP summarized general-to-specific usages, contexts, or phraseologies of words are quite effective in assisting learners in collocation and phrase learning.

5 Applying GRASP to Error Correction

To demonstrate the viability of GRASP-retrieved lexicalized grammar patterns (e.g., "*play ~ role* In V-ING" and "*look forward to* V-ING") in error detection and correction, we incorporate them into an extended Levenshtein algorithm (1966) to provide broad-coverage sentence-level grammatical edits (involving substitution, deletion, and insertion) to inappropriate word usages in learner text.

Previously, a number of interesting rule-based error detection/correction systems have been proposed for some specific error types such as article and preposition error (e.g., (Uria et al., 2009), (Lee et al., 2009), and some modules in (Gamon et al., 2009)). Statistical approaches, supervised or unsupervised, to grammar checking have become the recent trend. For example, unsupervised systems of (Chodorow and Leacock, 2000) and (Tsao and Wible, 2009) leverage word distributions in general and/or word-specific corpus for detecting erroneous usages while (Hermet et al., 2008) and (Gamon and Leacock, 2010) use Web as a corpus. On the other hand, supervised models, typically treating error detection/correction as a classification problem, utilize the training of well-formed texts ((De Felice and Pulman, 2008) and (Tetreault et al., 2010)), learner texts, or both pairwisely (Brockett et al., 2006). Moreover, (Sun et al., 2007) describes a way to construct a supervised error detection system trained on well-formed and learner texts neither pairwise nor error tagged.

In contrast to the previous work in grammar checking, our pattern grammar rules are automatically inferred from a general corpus (as described in Section 3) and helpful for correcting errors resulting from the others (e.g., "to close" in "play ~ role to close"), our pattern grammar lexicalizes on *both* content and function words and lexical items within may be contiguous (e.g., "*look forward to* V-ING PRP") or non-contiguous (e.g., "*play ~ role* In V-ING"), and, with word class (POS) information, error correction or grammatical suggestion is provided at sentence level.

5.1 Error Correcting Process

Figure 3 shows how we check grammaticality and provide suggestions for a given text with accurate spelling.

```
procedure GrammarChecking(T,PatternGrammarBank)
(1) Suggestions=""//candidate suggestions
(2) sentences=sentenceSplitting(T)
      for each sentence in sentences
(3)     userProposedUsages=extractUsage(sentence)
        for each userUsage in userProposedUsages
(4)       patGram=findPatternGrammar(userUsage.lexemes,
                                     PatternGrammarBank)
(5)       minEditedCost=SystemMax; minEditedSug=""
          for each pattern in patGram
(6)         cost=extendedLevenshtein(userUsage,pattern)
            if cost<minEditedCost
(7)           minEditedCost=cost; minEditedSug=pattern
          if minEditedCost>0
(8)         append (userUsage,minEditedSug) to Suggestions
(9) Return Suggestions
```

Figure 3. Procedure of grammar suggestion/correction.

In Step (1), we initiate a set *Suggestions* to collect grammar suggestions to the user text *T* according to a bank of patterns *PatternGrammarBank*, i.e., a collection of summaries of grammatical usages (e.g., "*play ~ role* In V-ING") of queries (e.g., "play role") submitted to GRASP. Since we focus on grammar checking at sentence level, *T* is heuristically split (Step (2)).

For each *sentence*, we extract user-proposed word usages (Step (3)), that is, the user grammatical contexts of ngram and collocation sequences. Take for example the (ungrammatical) sentences and their corresponding POS sequences "he/PRP play/VBP an/DT important/JJ roles/NNS to/TO close/VB this/DT deals/NNS" and "he/PRP looks/VBZ forward/RB to/TO hear/VB you/PRP". Ngram contexts include "*he* VBP DT", "*play an* JJ NNS", "*this* NNS" for the first sentence and "*look forward to* VB PRP" and "*look forward to hear* PRP" for the second. And collocation contexts for the first sentence are "*play ~ role to VERB*" and "*close ~ deal* ."

For each *userUsage* in the sentence (e.g., "*play ~ role* TO VB" and "*look forward to hear* PRP"), we first acquire the pattern grammar of its lexemes (e.g., "*play role*" and "*look forward to hear*") such as "*play ~ role in* V-ing" and "*look forward to hear from*" in Step (4), and we compare the user-proposed usage against the corresponding predominant, most likely more proper, ones (from Step (5) to (7)). We leverage an extended Levenshtein's algorithm in Figure 4 for usage comparison, i.e. error detection and correction, after setting up *minEditedCost* and *minEditedSug* for the minimum-cost edit from alleged error usage into appropriate one (Step (5)).

```
procedure extendedLevenshtein(userUsage,pattern)
(1) allocate and initialize costArray
      for i in range(len(userUsage))
        for j in range(len(pattern))
          //substitution
          if equal(userUsage[i],pattern[j])
(2a)        substiCost=costArray[i-1,j-1]+0
          elseif sameWordGroup(userUsage[i],pattern[j])
(2b)        substiCost=costArray[i-1,j-1]+0.5
          else
(2c)        substiCost=costArray[i-1,j-1]+1
          //deletion
          if equal(userUsage[i+1],pattern[j+1])
(3a)        delCost=costArray[i-1,j]+smallCost
          else
(3b)        delCost=costArray[i-1,j]+1
          //insertion
          if equal(userUsage[i+1],pattern[j+1])
(4a)        insCost=costArray[i,j-1]+smallCost
          else
(4b)        insCost=costArray[i,j-1]+1
(5)       costArray[i,j]=min(substiCost,delCost,insCost)
(6) Return costArray[len(userUsage),len(pattern)]
```

Figure 4. Extended Levenshtein algorithm for correction.

In Step (1) of the algorithm in Figure 4 we allocate and initialize *costArray* to gather the dynamic programming based cost to transform *userUsage* into a specific *pattern*. Afterwards, the algorithm defines the cost of performing substitution (Step (2)), deletion (Step (3)) and insertion (Step (4)) at i-indexed *userUsage* and j-indexed *pattern*. If the entries *userUsage*[*i*] and *pattern*[*j*] are equal literally (e.g., "VB" and "VB") or grammatically (e.g., "DT" and "PRP\$"[9]), no edit is needed, hence, no cost (Step (2a)). On the other hand, since learners tend to select wrong word form and preposition, we make less the cost of the substitution of the same word group, say from "VERB" to "V-ing", "TO" to "In" and "In" to "IN(*on*)" (Step (2b)) compared to a total edit (Step (2c)). In addition to the conventional deletion and insertion (Step (3b) and (4b) respectively), we look ahead to the elements *userUsage*[*i*+1] and *pattern*[*j*+1] considering the fact that "with or without preposition" and "transitive or intransitive verb" often puzzles EFL learners (Step (3a) and (4a)). Only a small edit cost is applied if the next elements in *userUsage* and *Pattern* are "equal". In Step (6) the extended Levenshtein's algorithm returns the minimum cost to edit *userUsage* based on *pattern*.

Once we obtain the costs to transform the *userUsage* into its related frequent patterns, we propose the minimum-cost one as its grammatical suggestion (Step (8) in Figure 3), if its minimum edit cost is greater than zero. Otherwise, the usage is considered valid. At last, the gathered suggestions *Suggestions* to *T* are returned to users (Step (9)). Example edits to the user text "*he play an important roles to close this deals. he looks forward to hear you.*" from our working prototype, EdIt[10], is shown in Figure 5. Note that we exploit context checking of collocations to cover longer span than ngrams', and longer ngrams like fourgrams and fivegrams to (more or less) help semantic checking (or word sense disambiguation). For example, "*hear*" may be transitive or intransitive, but, in the context of "*look forward to*", there is strong tendency it is used intransitively and follows by "*from*", as EdIt would suggest (see Figure 5).

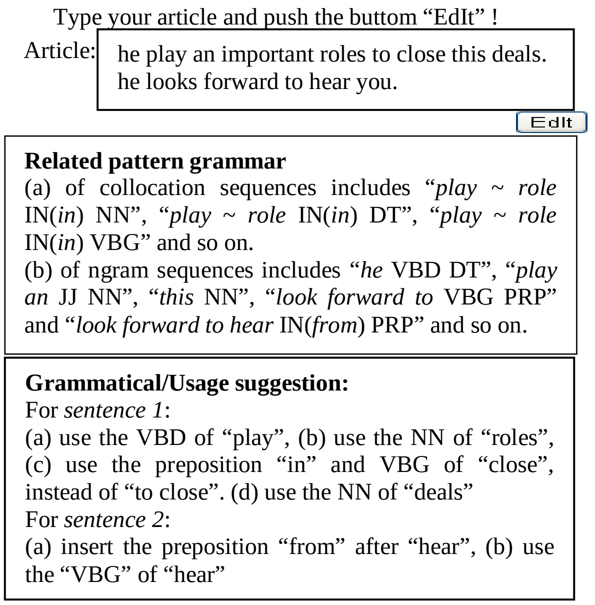
There are two issues worth mentioning on the development of EdIt. First, grammar checkers typically have different modules examining different types of errors with different priority. In our unified framework, we set the priority of checking collocations' usages higher than that of ngrams', set the priority of checking longer ngrams' usages higher than that of shorter, and we do not double check. Alternatively, one may first check usages of all sorts and employ majority voting to determine the grammaticality of a sentence. Second, we further incorporate

[9] ONE'S denotes possessives.

[10] http://140.114.214.80/theSite/EdIt_demo2/

Erroneous sentence	EdIt suggestion	ESL Assistant suggestion
Wrong word form		
… a sunny days …	a sunny NN	a sunny day
every days, I …	every NN	every day
I would said to …	would VB	would say
he play a …	he VBD	none
… should have tell the truth	should have VBN	should have to tell
… look forward to see you	look forward to VBG	none
… in an attempt to seeing you	an attempt to VB	none
… be able to solved this problem	able to VB	none
Wrong preposition		
he plays an important role to close …	play ~ role IN(in)	none
he has a vital effect at her.	have ~ effect IN(on)	effect on her
it has an effect on reducing …	have ~ effect IN(of) VBG	none
… depend of the scholarship	depend IN(on)	depend on
Confusion between intransitive and transitive verb		
he listens the music.	missing "to" after "listens"	missing "to" after "listens"
it affects to his decision.	unnecessary "to"	unnecessary "to"
I understand about the situation.	unnecessary "about"	unnecessary "about"
we would like to discuss about this matter.	unnecessary "about"	unnecessary "about"
Mixture		
she play an important roles to close this deals.	she VBD; an JJ NN; play ~ role IN(in) VBG; this NN	play an important role; close this deal
I look forward to hear you.	look forward to VBG; missing "from" after "hear"	none

Table 2. Three common score-related error types and their examples with suggestions from EdIt and ESL Assistant.


Type your article and push the buttom "EdIt" !

Article: he play an important roles to close this deals. he looks forward to hear you.

EdIt

Related pattern grammar
(a) of collocation sequences includes "*play ~ role* IN(*in*) NN", "*play ~ role* IN(*in*) DT", "*play ~ role* IN(*in*) VBG" and so on.
(b) of ngram sequences includes "*he* VBD DT", "*play an* JJ NN", "*this* NN", "*look forward to* VBG PRP" and "*look forward to hear* IN(*from*) PRP" and so on.

Grammatical/Usage suggestion:
For *sentence 1*:
(a) use the VBD of "play", (b) use the NN of "roles", (c) use the preposition "in" and VBG of "close", instead of "to close". (d) use the NN of "deals"
For *sentence 2*:
(a) insert the preposition "from" after "hear", (b) use the "VBG" of "hear"

Figure 5. Example EdIt responses to the ungrammatical.

probabilities conditioned on word positions to weigh edit costs. For example, the conditional probability of "VERB" being the immediate follower of "look forward to" is virtually zero, but the probability of "V-ing" is around 0.3.

5.2 Preliminary Results in Error Correction

We examined three common error types in learner text that are highly correlated with essay scores (Leacock and Chodorow, 2003; Burstein et al., 2004), to evaluate EdIt, (see Table 2). In Table 2, the results of a state-of-the-art checker, ESL Assistant (www.eslassistant.com/), are shown for comparison, and information produced by both systems are underscored. As indicated, GRASP retrieves patterns which are potential useful if incorporated into an extension of Levenshtein's algorithm to correct substitution, deletion, and insertion errors in learner.

6 Summary

We have introduced a new method for producing a general-to-specific usage summary of the contexts of a linguistic search query aimed at accelerating learners' grasp on word usages. We have implemented and evaluated the method as applied to collocation and phrase learning and grammar checking. In the preliminary evaluations we show that GRASP is more helpful than traditional language learning tools, and that the patterns and lexical bundles provided are promising in detecting and correcting common types of errors in learner writing.

References

Morton Benson, Evellyn Benson, and Robert Ilson. 1986. *The BBI Combinatory Dictionary of English: A* *guide to word combinations*. Philadelphia: John Benjamins.

Chris Brockett, William B. Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the ACL*, pages 249-256.

Jill Burstein, Martin Chodorow, and Claudia Leacock. 2004. Automated essay evaluation: the criterion online writing service. *AI Magazine*, 25(3): 27-36.

Yu-Chia Chang, Jason S. Chang, Hao-Jan Chen, and Hsien-Chin Liou. 2008. An automatic collocation writing assistant for Taiwanese EFL learners: a case of corpus-based NLP technology. *CALL*, 21(3): 283-299.

Winnie Cheng, Chris Greaves, and Martin Warren. 2006. From n-gram to skipgram to concgram. *Corpus Linguistics*, 11(4): 411-433.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of the NAACL*, pages 140-147.

Rachele De Felice and Stephen G. Pulman. 2008. A classifer-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the COLING*, pages 169-176.

Philip Durrant. 2009. Investigating the viability of a collocation list for students of English for academic purposes. *ESP*, 28(3): 157-169.

John R. Firth. 1957. Modes of meaning. In *Papers in Linguistics*. London: Oxford University Press, pages 190-215.

Michael Gamon, Claudia Leacock, Chris Brockett, William B. Dolan., Jianfeng Gao, Dmitriy Belenko, and Alexandre Klementiev. 2009. Using statistical techniques and web search to correct ESL errors. *CALICO*, 26(3): 491-511.

Michael Gamon and Claudia Leacock. 2010. Search right and thou shalt find … using web queries for learner error detection. In *Proceedings of the NAACL*.

Matthieu Hermet, Alain Desilets, and Stan Szpakowicz. 2008. Using the web as a linguistic resource to automatically correct lexico-syntatic errors. In *Proceedings of the LREC*, pages 874-878.

Susan Hunston and Gill Francis. 2000. *Pattern Grammar: A Corpus-Driven Approach to the Lexical Grammar of English*. Amsterdam: John Benjamins.

Jia-Yan Jian, Yu-Chia Chang, and Jason S. Chang. 2004. TANGO: Bilingual collocational concordancer. In *ACL Poster*.

Adam Kilgarriff, Pavel Rychly, Pavel Smrz, and David Tugwell. 2004. The sketch engine. In *Proceedings of the EURALEX*, pages 105-116.

Chong Min Lee, Soojeong Eom, and Markus Dickinson. 2009. Toward analyzing Korean learner particles. In *CALICO Workshop*.

Claudia Leacock and Martin Chodorow. 2003. Automated grammatical error detection. In M.D. Shermis and J.C. Burstein, editors, *Automated Essay Scoring: A Cross-Disciplinary Perspective*, pages 195-207.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. Soviet Physics Doklady, 10, page 707.

Diane Nicholls. 1999. The Cambridge Learner Corpus – error coding and analysis for writing dictionaries and other books for English Learners.

John M. Sinclair. 1987. The nature of the evidence. In J. Sinclair (ed.) *Looking Up*. Collins: 150-159.

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

Michael Stubbs. 2004. At http://web.archive.org/web/20070828004603/http://www.uni-trier.de/uni/fb2/anglistik/Projekte/stubbs/icame-2004.htm.

Guihua Sun, Xiaohua Liu, Gao Cong, Ming Zhou, Zhongyang Xiong, John Lee, and Chin-Yew Lin. 2007. Detecting erroneous sentences using automatically mined sequential patterns. In *Proceedings of the ACL*, pages 81-88.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for prepositions selection and error detection. In *Proceedings of the ACL*, pages 353-358.

Nai-Lung Tsao and David Wible. 2009. A method for unsupervised broad-coverage lexical error detection and correction. In *NAACL Workshop*, pages 51-54.

Larraitz Uria, Bertol Arrieta, Arantza D. De Ilarraza, Montse Maritxalar, and Maite Oronoz. 2009. Determiner errors in Basque: analysis and automatic detection. *Procesamiento del Lenguaje Natural*, pages 41-48.

David Wible and Nai-Lung Tsao. 2010. StringNet as a computational resource for discovering and investigating linguistic constructions. In *NAACL Workshop*, pages 25-31.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the ACL*, pages 189-196.