The forthcoming issue of *ReCALL* (Vol. 17, Part 2) will be distributed to EUROCALL members in November/December 2005. Please send articles, software reviews, details of relevant events or other items of interest for future issues to June Thompson, Editor *ReCALL*  d.j.thompson@hull.ac.uk.

The journal contents are listed at:

http://www.upv.es/eurocall/r_contents.htm

All articles are considered by an international panel of referees. Notes for contributors can be found at:

http://www.upv.es/eurocall/ contrib.htm

---

**Projects**

**LIPSTIC**

A Limited Intelligence Parser Seeking Typical Interference Constructions

**Introduction**

Several people have asked me why I embark on a major project for trapping errors in learners’ free texts when there is a perfectly good Grammar Checker in Microsoft Word. Unfortunately, the Word Grammar Checker is ill-suited to learner English. I put the following text through it and it failed completely.

*My friends who last time saw you wants that you come soon. Do you can get very early next week the bus because we must you see? I not see you for very long time.*

In contrast, the errors that LIPSTIC will pick up and point out here are the following:

1: You cannot put adverbials of time like "last time" between a subject ("who") and the following verb.
2: The expression "my friends" is plural, but "wants" is 3rd singular.
3: After "want" you cannot have "that" and a finite clause.
4: You cannot use "can" after "do".
5: The expression "the bus" should come next to "get".
6: "You" should come after "see".
7: An auxiliary verb is needed in "I not see".

It probably will not be able to tell me that I must use the perfect auxiliary for "I not see you". However, the fact that there is "for" followed by a recognisable time expression means that it could suggest that the perfect may be required here. It definitely will not be able to suggest that "would like" would be more appropriate than "must".

The Word Grammar Checker accepted this text except for one thing. It told me that if I use "bus", then later on I should use "sees", or alternatively, change "bus" to "buses".
It has a bad reputation, and not only for learner English, as you will see if you check this website: http://faculty.washington.edu/sandeep/check/

In recent years there have been papers at EUROCALL conferences on parsing, error trapping, mistake catching, or whatever you would like to call it. Many of them were concerned with the major task of parsing English sentences, and then adding procedures which would also be able to continue to parse the sentence if a sentence violated the grammatical rules of English, with the ultimate aim of giving the learner some kind of feedback. It is certainly not my wish to criticise these various excellent projects, but I have not yet seen one which was going to be usable for my goals. I want to have a tool which will look at a freely written text by a learner, in the first instance a German learner. Taking into account interference errors based on other languages and the large number of cross-linguistic typical errors, it would be possible to adapt it later on to learners of other learner backgrounds. It will look at the text sentence by sentence, check it and see if it can say anything helpful about mistakes in the grammar. This tool would be usable either within a web-based or CD-based language teaching program, or could also function as an Add-In for Microsoft Word.

My disappointment with projects I have seen to date is that they seemed to be nowhere near ready for real use. And, in all humility, I must say that the same goes for my own contribution. The reason I wish to discuss it now is that I need feedback and ideas from other researchers who are interested in this topic. So my aim here is to describe my program, set it in the context of its future use, explain a little about how I intend to approach various issues, and show how the provisional User Interface works.

My belief is that for what I want to do, short cuts rather than full-blown parsing can achieve results. I would like to draw your attention to the acronym in the title of my article, LIPSTIC, which stands for a "Limited Intelligence Parser Seeking Typical Interference Constructions". The acronym is there to pull it all together, but the real title makes a number of claims which describe how I have been going about this project. It has "limited intelligence", by which I mean that it does not attempt to be successful in parsing all sentences, in finding all the errors in every sentence, or even giving any feedback at all on every sentence. But it will undoubtedly identify quite a number of grammatical errors written in a free text. And those who use it will be aware that it identifies some errors, but not all. Clearly it will not be able to deal with semantic oddities, and its feedback on use of tense will be limited to an attempt to see whether past or perfect has been correctly selected.

It does, of necessity, have to partially parse a sentence, to identify subjects and verb complements, and much else. Otherwise it could do nothing. But once it has found major sentence elements (assuming that it is usually successful in that), it then abandons the elusive complete parse, and starts searching for typical interference errors. My aim here is to give some insight into how it deals with text pre-editing, lexical assignment, preliminary reduction of lexical ambiguity, further reduction of ambiguity, template application and appropriate feedback levels.

**Text pre-editing**

The first problem that one encounters is that learners' texts will not be the kinds of texts that we would like to have as input to our procedures. For example, one may hope that one will identify a sentence by a sentence-final punctuation mark. But a typical text will often begin with a title, or if it is a letter, an address, with telephone numbers. Learners may punctuate wrongly, leaving a space between word and punctuation mark, use inverted commas carelessly, use strange signs or characters which our search mechanisms cannot cope with. Well, this is simple but necessary housekeeping. In an example text, the following sentence is encountered by the **Mistake Catcher**. The Hands and Labels are interesting, very interesting!
And then the following messages are delivered in turn, after the User has made the necessary correction:

1. Surely the first word of the sentence should have a capital letter?
2. You have a space before a punctuation mark. Please close ",," up to the word before it.
3. There are two words in the sentence (not the first) beginning with a capital letter. Are they names? If NO, please make the capitals into small letters.

Learner texts are full of such surface level irregularities, and these are just a few simple examples of the kind of pre-editing which is undertaken. The first paragraph, which appears in the topmost window of the tool below, contains odd symbols, wrongly placed inverted commas, numerals joined up to words, etc. These surface phenomena are worked through for each sentence before the parsing and real grammatical mistake catching takes place.

An explanation of this screenshot would perhaps be helpful. The whole text which has been written by the User appears in the upper window. The sentence currently being processed appears below it in red. Feedback messages appear in the window below that, as well as the operator button. In the first shaded box below the feedback message, the corrected text appears. Note that the title "Example Text" is copied in automatically from the topmost box because it is paragraph-like, but without punctuation, and is assumed to be a title, part of an address in a letter, or some other such thing. This is an important facility when processing typical texts which are learners are likely to produce. Finally, at the very bottom there is a table with numbers, which is
there only as feedback for the developer, and can be turned off by clicking the relevant box at the bottom. This interface is only for development purposes, and does not represent what the final interface will be like.

**Lexical Assignment**

One of the reasons that my immediate goal is to provide this tool for intermediate learners of English is that I want to manage with a dictionary of about five thousand words. Apart from the huge task of relatively standard grammatical coding for these words, extensive attention has to be paid to the coding of verbs with respect to the complements that they can have, or in the terminology of the OALD, the verb patterns that can be associated with each of these verbs.

Another really important reason for restricting LIPSTIC in the first instance to intermediate learners is that the complexity of the sentences that they write is fairly limited. Although I am building LIPSTIC so that it will be able to deal with higher levels of complexity, e.g. multiply embedded relative clauses, I don't think I have to worry about more than one level of embedding in the first instance.

Obviously, I have a specific goal in mind, to deal with learner English, and to this end, I have divided up the dictionary two, or even three ways. The first dictionary contains all parts of speech, but excludes all the base regular forms of nouns, verbs and adjectives. So it looks for exact matches in the sentence under consideration. Words which it does not find, and also words which are irregular forms (like put as past tense), but where the form is also a regular (present or infinitive) form, are put into a new search list for a pass through the second dictionary. And the second dictionary only contains the base form of each word, so we do not have lots of multiple entries. The interaction of these two dictionaries makes for a very economical dictionary search.

Common misspellings or common error forms can also be included in the first dictionary search, so that more informative feedback can be given for such errors than would normally be provided by a spelling checker. So, for the lexeme "put", the following forms occur in the first dictionary, which searches for exact matches:

- put#98 (Participle)
- put#99 (Past Tense)
- putted#100 (predicted learner error)

If *put* is encountered in the sentence being processed, forms 98 and 99 are entered into the active parse, but a code attached to these entries indicates that the dictionary search for instances of the lexeme "put" should not yet be considered as complete. So when the program proceeds to the second dictionary search, it will encounter another instance of put:

- put#33 (base or infinitive form)

If any of the following regular forms are encountered in a sentence, they will be attached to the lexeme "put", together with the appropriate coding for their grammatical characteristics: *put, puts, putting*. If the form *put* is found in this second lexical pass, it will be a third lexical entry for this word, because the form *put* will already have been entered as being potentially a past tense form or a past participle when the first dictionary was consulted.

Finally, a word on the Proper Names dictionary. It makes sense to have one because we have proper names for time expressions which I need to recognise, like days of the week and months of the year. For common first names, it will usually allow us to know what kind of pronouns should relate to it, although I don't think we will be able to do much with gender. In addition, any words beginning with a capital letter will be classified as Proper Names, so they don't have to be in the dictionary. I should mention that this only happens after a special warning message for German learners. We have to make sure that they have not mistakenly written a common noun with a capital letter.

Sometimes learners will either misspell words in ways not predicted in the dictionary. I originally planned to have a spelling checker go over the whole text before the individual sentences were checked, but I think it makes more didactic sense to deal with them as apparently unknown words, on a sentence by sentence basis, in context, and ask the learner to check whether the spelling is correct. This is above all intended to be
a learning tool, and not just an automatic correcting tool. A whole series of rapid learner
decisions about spelling mistakes via a spelling checker is not very helpful didactically.
Another argument in favour of not using a conventional spelling checker, already
mentioned above, is that in cases where we have predicted a common German-based
spelling error, we can give a useful feedback message on it.

**Dealing with lexical ambiguity and reducing structures**

Words which seem totally unambiguous to us in context are often multiply ambiguous
from the lexical point of view. Take the apparently harmless sentence:

*She let me put a bandage on her cut.*

The numbers below the underlined words indicate how many different grammatical
forms this spelling represents. Multiplying these together, we see that there are 600
possible analyses of this sentence. Many of them will be ruled out as soon as we start
trying to parse it. But if a nine word sentence can have so many variants, we would
have a huge parsing task, so for practical reasons I try to reduce the number of
ambiguities at the very time that I am building my parsing grid. Thus, although
"bandage" is in the dictionary as either a finite or non-finite lexical verb, the fact that it
is also in the dictionary as a noun and here in this sentence is adjacent to an article
allows me NOT to build structures with "bandage" as a verb, even before I start. You
will see some examples of this now, and how it can already lead to useful error
messages at the pre-parsing stage. And I stress that these are NOT meant to be
examples of German English errors, but are simply sentences which have been thought
up to check various aspects of lexical search and pre-parsing.

It looks at the sentence *The manages has managed*. With the aim of reducing
impossible structures, the pre-parsing analysis, which works by looking simply at
contiguous words, rather than words in a structural relationship, notices
that *manages* which can only be a third singular verb directly follows an article, and
gives the following message:

1. It looks as if you have put the verb "manages" directly after an article or
determiner. This can't be right. Please change it.

Preliminary analysis also deals with predicted spelling errors, so when it encounters the
word *refering* in *Is the man refering to the hand?*, it points out:

2: When the second syllable is stressed, as in "refer", you have to double the final
're' before you add the ending. Please correct it.

And its examination of contiguous words leads it to comment on *James was breathing
and still does breathes* as follows:

3: It looks as if you have put a finite verb, "breathes", after an auxiliary or helping
verb, "does". This can't be right. Please change it.

I could go on giving similar instances from the Example Text, but the point should be
clear. It is essential to limit as much as possible the number of potential sentences that
the *Mistake Catcher* will have to parse, so we have a device here which eliminates a
large number of unacceptable variants even before we build up the parsing grid.

**How to deal with remaining ambiguity**

Let's call what I have just discussed with reference to "bandage", and demonstrated
with other examples: 'adjacency parsing'. It is quite reliable, and its purpose is simply
to get rid of obviously unnecessary structures. However, I calculate that even using that
technique, I could still have up to 100 possible analyses of a fairly simple sentence. How
do I deal with that? Well, it is simple and difficult.

There is a whole series of parsing and error trapping procedures to be applied one after
the other to each version of the sentence. If the first procedure gets stuck on some of
them, but not all of them, those that it gets stuck on are deleted. And we then move on
to the next procedure on all the remaining alternatives.
What happens if it gets stuck on all of them? Well, it will give a feedback message on the first (or any one of them), however, it may be necessary to decide between possible feedback messages, i.e. one may be too severe for the immediate environment, e.g. "You have used a noun and you need a verb here", and the other more accurate and relevant: "You have used the wrong form of the verb here". The fairly difficult task in such a situation is to select the best message. But the really important thing is that even if the best feedback message isn't chosen, LIPSTIC will have identified a point where something needs to be changed.

This kind of potential ambiguity amongst various errors is probably the most difficult thing that I have to deal with in the coming months.

**Application of templates with examples**

One of the most common learner errors which we are familiar with is inserting some kind of adverbial between the verb and the object. If the verb takes an obligatory object, the Mistake Catcher will search for a likely candidate, then examine anything which comes between the lexical verb and that object, suggesting that the object string immediately follow the verb. But it will also count the words in the object string, and if they significantly exceed in number the intervening words, it will assume that the User has corrected postponed a heavy object, and give no message. An example of this is the following sentence, where the short item intervening between the verb and the long object is *yesterday*.

> I phoned yesterday the boy who offered to repair my computer.

It should be pointed out that the correct recognition of time expressions like *last week*, is absolutely essential for catching mistakes. Compare these sentences, where last week could be an object and adverbial in i, only an object in iii and iv, and is ungrammatical on account of its position only in iv:

i: I remembered last week.
ii: I remembered her birthday
iii: I remembered her birthday last week.
iv: *I remembered last week her birthday.

The verbal part of the sentence is, even in English, the source of many morphemic or grammatical errors, but the structure of the auxiliary phrase is extremely rigid, and will not present many difficulties. Needless to say, we have to identify subjects efficiently in order to deal with questions where one verbal element is separated from the rest by the subject, which may also contain a verbal element in a relative clause, and we have to make due allowances for intervening adverbials.

I aim to deal with floating quantifiers, because quantifiers often float in situations where it is not permitted in learner English, e.g. iii and iv below.

i: All the girls have arrived.
ii: The girls have all arrived.
iii: *The girls have arrived all.
iv: *The girls all have arrived.

This phenomenon will only occur with relation to words such as "all" or "both", and is a good example of where my "template search" will identify common errors.

The position of adverbs will be difficult to give helpful feedback on, except in some well-defined positions where any kind of adverb will be unacceptable. However, the position of frequency adverbs (e.g. "usually", "never") is extremely fixed in English. There is a limited number of them, and they are a frequent source of word order errors. Again the template based approach will work well here.

Finally, with respect to the identification of mistakes, an error-based "phrase matching" lexicon is planned. What I mean by this is simply a search procedure which will look for what are almost always unacceptable phrases in English, but are often written by learners on the basis of transfer from their own language. An obvious one would be the use of "on" rather than "at", as for example in "She's on the grammar school" rather than "She's at the grammar school". A good phrasal error database will provide opportunities for feedback on this kind of error.
I do not wish to underestimate the task of catching errors. But I hope that this gives some insight into the approach I am adopting. Finally, a word on feedback.

**Pedagogical aspects**
The examples presented above in a number of feedback messages are not intended to be evaluated here at his point. They were written to make things clearer for the developer. I have nevertheless tried to write them in a user-friendly style. I plan to have at least two levels of feedback, one for those who know common grammatical terms, and one for those to whom such terminology is arcane. But the real question to conclude with is this: To what extent will such an Error Trapper help people in writing free texts?

There is always the danger that it will give them a false sense of security, because it will undoubtedly accept lots of unacceptable sentences. But it has the advantage of giving the User immediate feedback on lots of errors. It works at the press of a button, and it helps learners to learn. I do not really wish to use the "A" word, "autonomy", which has been so dominant in EuroCALL as in so many other academic groups in recent years, but perhaps I should. It makes the learner more independent, more autonomous. When he or she has to write a text, they will get useful information immediately, without a teacher intervening, and will be in control of making decisions about whether to accept or reject proposals. And I think it will help learners. Speaking for myself, I live and work in Germany, and would love to have such a tool for when I have to try to write in German.

*James Pankhurst*

Freelance software author and Editor Emeritus of Second Language Research