A NEW METHOD FOR PARSING STUDENT TEXT TO SUPPORT COMPUTER-ASSISTED ASSESEMENT OF FREE TEXT ANSWERS

Elizabeth Guest and Sally Brown

A New Method for Parsing Student Text to Support Computer-Assisted Assessment of Free Text Answers

Elizabeth Guest and Sally Brown, Leeds Metropolitan University

Abstract

Due to current trends in staff-student ratios, the assessment burden on staff will increase unless either students are assessed less, or alternative approaches are used. Much research and effort has been aimed at automated assessment but to date the most reliable method is to use variations of multiple choice questions. However, it is hard and time consuming to design sets of questions that foster deep learning. Although methods for assessing free text answers have been proposed, these are not very reliable because they either involve pattern matching or the analysis of frequencies in a "bag of words".

The first step towards automatic marking of free text answers by comparing the meaning of student answers with a single model answer is to parse the student work. However, because not all students are good at writing grammatically correct English, it is vital that any parsing algorithm can handle ungrammatical text. In this paper, we present preliminary results of using a relatively new linguistic theory, Role and Reference Grammar, to parse student texts and show that ungrammatical sentences can be parsed.

Introduction

In the current climate of increasing student numbers and decreased funding per student in many HEIs internationally, it is necessary to find economies of scale in teaching and supporting undergraduate students. Economies of scale are possible to a certain extent for lectures and tutorials, but this is less possible for assessment. As staff student ratios decrease, the assessment burden on staff will increase unless alternative approaches are used.

One solution to this dilemma is to seek ways to mark student work automatically. This is being done at present using variations on multiple choice questions, with a variety of innovative question types that test learning beyond simple recall. If designed correctly, these kinds of tests can provide students with immediate feedback on how well they are doing and can provide valuable formative pointers for further learning. Extensive evidence demonstrates that increased formative assessment can impact positively on student learning and retention (Sadler, 1989) (Sadler, 1998) (Rust, 2002) (Sambell and Hubbard, 2004) (Yorke, 2001). However, it can be difficult to design if we want to make it truly an integral part of learning and if we want to avoid encouraging inappropriate student behaviour, such as random guessing of answers.

Considerable work has been undertaken in recent years to investigate and implement approaches to CAA that foster deep learning (Beevers et al., 1989) (Brown et al., 1999), but significant advances still remain to be made. Some have argued that it is currently possible to assess essays by automatic means but we remain unconvinced. However, it would be very helpful if it were possible to automatically mark short free text answers using CAA approaches, thus reducing the drudgery for markers. This would allow more scope in the setting of questions and would give students more opportunity to show what they understand and can do. Much research has been aimed at this question, but this generally either involves pattern matching (Sukkarieh et al., 2003) (Sukkarieh et al., 2004) or latent semantic analysis (Wiemer-Hastings, 2001) (Landauer et al., 1997), or a combination of these (Pérez and Alfonsa, 2005). These methods work to a certain extent, but because they are not based on the meaning of the text, they are quite easy to fool. For instance latent semantic analysis can be fooled by writing down the right kinds of words in any order. The problem with current approaches to pattern matching on the other hand, is that if the student writes down a correct answer in a different way, it will be marked wrong.

Our innovative approach is based on the grammatical tradition of parsing, that is breaking down language into its functional components like verbs, nouns and adverbs. Role and Reference Grammar (RRG) (Van Valin and LaPolla, 1997) (Van Valin, 2005) is a relatively new linguistic theory that majors on predicates and their arguments. It separates the most vital parts of the sentence from the modifiers (adverbs, adjectives, auxiliaries, and articles). This means that the core meaning can be extracted first and then the modifiers fitted in at a later stage. As long as the arguments and the verbs are in the correct order for English (subject verb object) then the sentence can be understood. It doesn't matter if (for example) Chinese students forget the articles, the sentence can still be parsed and the meaning extracted. The core meaning of the sentence is extracted via the use of templates. This makes it easier to extract the important parts of the meaning of the sentence: we just need to identify the predicate and the arguments which are clearly labelled branches within the templates.

In this work we describe a method for using the RRG paradigm for parsing student texts, which do not have to be grammatically correct. This work can be used as a pre-processing step to those methods that use latent semantic analysis or pattern matching. There is evidence to suggest that latent semantic analysis gives better results when the subject, verb, and object of the sentence is used rather than an unstructured "bag of words" (Wiemer-Hastings, 2001). Our method will provide a mechanism for extracting some structure. If structure can be extracted, then this structure can also be passed to a pattern matcher, which will decrease the number of possibilities that have to be included. This method will also enable accurate marking of ungrammatical sentences.

Parsing for Role and Reference Grammar

Role and Reference Grammar (RRG) (Van Valin and LaPolla, 1997) (Van Valin, 2005) was developed as a result of asking the question “What would a linguistic theory look like if it was based on Lakhota and Tagalog rather than English?”. The result is a theory that is suited to describe a huge range of languages, including English. Of all the linguistic theories, it is most closely related to functional grammar, but there are important differences.

Role and Reference Grammar posits algorithms to go from syntax to semantics and semantics to syntax. The main contribution is the use of parsing templates and the notion of the CORE. A CORE consists of a predicate (generally a verb) and (normally) a number of arguments. It must have a predicate. Everything else is built around one or more COREs. Simple sentences contain a single CORE; complex sentences contain several COREs.

The fact that RRG focuses on COREs, means that the semantics is relatively easy to extract from a parse tree. You just have to look for the PRED, and ARG branches of the CORE to obtain the predicate (PRED) and the arguments (ARG). Who did what to whom will depend either on the ordering of the ARG branches (in the case of English), or on their cases, or both.

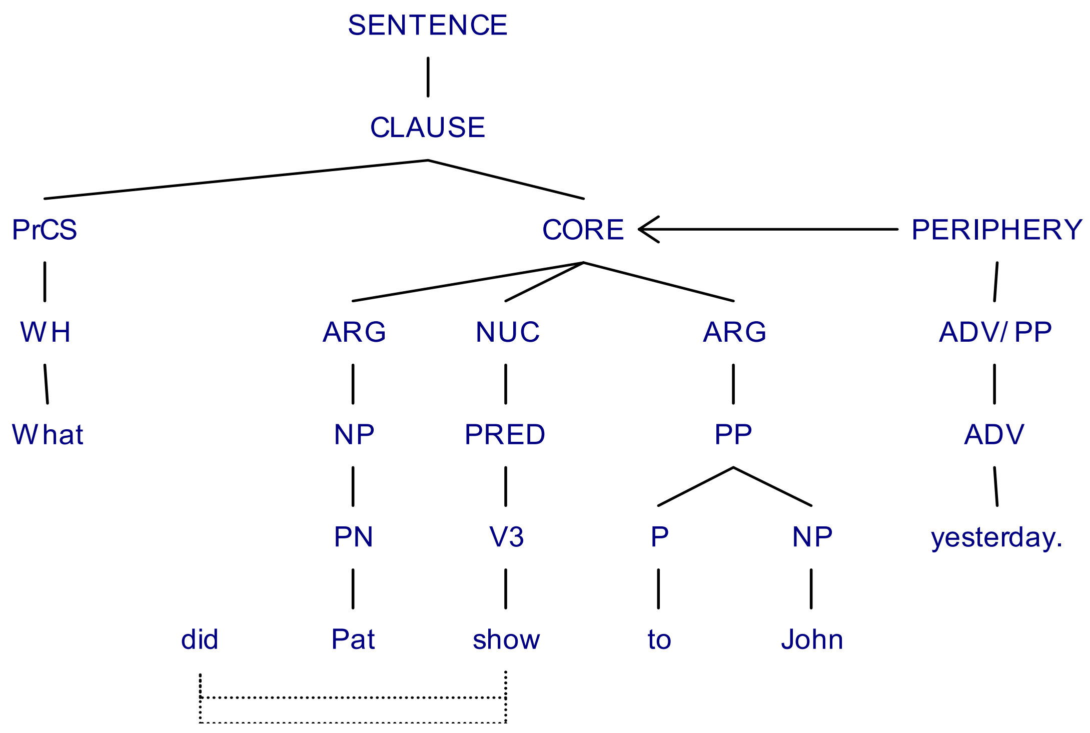


Figure 1: Example RRG parse tree.

An example of an RRG parse tree is given in figure 1. Notice that in this example, the word “did” does not feature in the parse tree, but it is linked to the verb “show”. This is because it is an operator. An important feature of

RRG from a parsing point of view is that parsing happens in two projections: the constituent projection, shown in figure 1 and the operator projection, which consists of words which modify other words (such as auxiliaries and adjectives). This is important because modifiers are often optional and it simplifies the parsing process considerably if these can be handled separately. Note that adverbs, which can modify larger constituents (such as COREs and CLAUSEs) go in the constituent projection so that it is clear what they are modifying. “Yesterday” in this example is an adverb which modifies the CORE, to show when the action took place.

RRG makes extensive use of templates. These templates consist of whole trees and are thus harder to use in a parsing algorithm than rules. The templates can easily be reduced to rules, but only at a loss of much important information. The example in figure 1 consists of one large template that gives the overall structure and some simple templates (which are equivalent to rules) so that elements such as NP and PP can be expanded. An NP is a noun phrase and in this theory consists of a noun, pronoun, or question word. Templates are required to parse complex noun phrases, such as those with embedded clauses. A PP is a prepositional phrase and consists of a preposition followed by a NP. Clearly if we reduce the template in the example in figure 1 to the rule

CLAUSE → NP NP V PP ADV

we lose a lot of the information inherent in the structure of the template. A further feature of RRG is that the branches of the templates do not have to have a fixed order and lines are allowed to cross. The latter is important for languages such as German and Dutch where the adverb that makes up the periphery normally occurs within the core. This feature will be important in our application for marking work by students for whom English is not their first language.

The above features pose challenges for parsing according to the RRG paradigm. We have overcome these challenges by making some additions to the standard chart parsing algorithm. The main innovations are

a) a modification to enable parsing with templates
b) a modification to allow variable word order.

In addition, parsing also includes elements of dependency grammar to find operators and to determine which word they belong to. At present the most popular methods of parsing are HPSG (Hou and Cercone, 2001, Kešelj, 2001, Wahlster, 2000) and dependency grammar (Chung and Rim, 2004, Covington, 2003, Holan, 2002). HPSG is good for fixed word order languages and dependency grammar is good for free word order languages. The approach to parsing described below is novel in that is allows parsing with templates, and because of the range in flexibility of word order allowed.

Outline of the parsing algorithm

The parsing algorithm relies on correctly tagged text. We use Shoebox (available from SIL (www.sil.org/computing/shoebox)) to tag sentences. Shoebox is a semi-interactive tagging program. It was chosen because the user can define their own tags and because it is easy to ensure all tags are correct. This is a good program to use for experimentation. Once the tags have been finalised an appropriate automatic tagger can be used, or written using standard techniques.

Once a sentence has been tagged, there are three parts to the parsing algorithm:

1. **Strip the operators.** This part removes all words that modify other words. It is based on a correct tagging of head and modifying words. This stage uses methods from dependency grammar and the end result is a simplified sentence.

2. **Parse the simplified sentence using templates.** This is done by collapsing the templates to rules, parsing using a chart parser and then rebuilding the trees at the end using a complex manipulation of pointers. The chart parser has been modified to handle varying degrees of word order flexibility. This is done by working out all the possible combinations of the ordering using breadth first search. These options are then built into a complex data structure in such a way that relevant parts are deleted as parsing progresses, leaving the correct option according to the data.

3. **Draw the resulting parse tree.**

Details of the extensions to the chart parser are given below.

Parsing Templates

Templates are parsed by collapsing all the templates to rules and then re-building the correct parse tree once parsing is complete. This is done by including the template tree in the rule, as well as the left and right hand sides. When rules are combined during parsing, we make sure that the right hand side elements of the instantiated rule, as represented in the partial parse tree, point to the leaves of the appropriate rule template tree. This is especially important when the order of the leaves of the template may have been changed. The reference number for the rule that has been applied is also recorded so that it can be found quickly.

Modifying nodes, such as PERIPHERY, cause problems with rebuilding the tree. This is because such nodes can occur anywhere within the template, including at the root and leaf levels. Also, if we are dealing with a sub-rule whose root node in the parse tree has a modifying node, it is not possible to tell whether this is a hang-over from the previous template, or part of the new template. To solve this problem, modifying nodes have flags to say whether they have been considered or not. There is a potential additional problem with repeated nested rules because if processing is done in the wrong order, the pointers to the rule template tree get messed up. To overcome this problem, each leaf of a template is dealt with before considering sub-rules.

The algorithm for building the tree is:

1. Get the appropriate rule and rule template tree
2. If the rule tree is of depth 1 and has no embedded modifying nodes (that is modifying nodes that point to a node other than the root), then we can simply continue by looking at each of the children in turn, starting at step 1.
3. If the rule tree is of depth greater than 1 or there are embedded modifying nodes, then make the rule template tree point to the appropriate places in the parse tree. This is done using the links made from the parse tree to the rule template tree during parsing. Note that the parse tree will consist of simple rule structures of depth 1 and modifying nodes will show up as children.
4. Clear all the children in the parse tree. This will have the effect of removing any embedded modifying nodes.
5. Copy all the children of the template tree and copy into the appropriate place in the parse tree.
6. If the template has modifying nodes, copy that part of the template tree and insert into the appropriate place in the parse tree.
7. Replace the leaves of the copied template trees with the original leaves. This is possible because the template leaves are pointing to the original leaves (step 3).
8. Consider each leaf in turn, modifying the parse tree as above (start at step 1 for each leaf).

Parsing with fixed, free, and constrained word order

There were two main problems to solve in order to modify the chart parser to handle varying degrees of word order flexibility:

1. Working out a notation for denoting how the word order can be modified.
2. Working out a method of parsing using this notation.

(1) was achieved by the following notation on the ordering of the leaves of the template, treating the template as a rule.

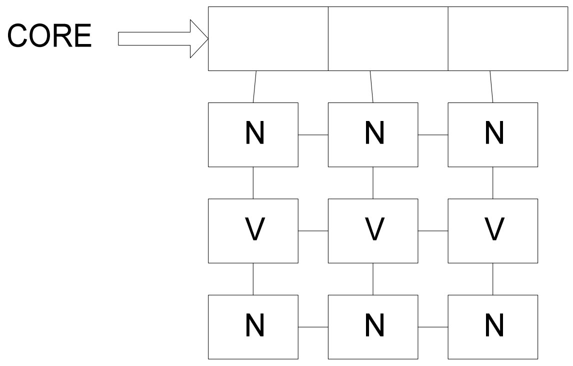
- Fixed word order: leave as it is {N V N}
- Free word order: insert commas between each element {N,V,N} (Note that case information is included as an operator so that the undergoer and actor can be identified once parsing is complete.)
- An element has to appear in a fixed position: use angular brackets: {N, <V>, ADV} this means that N and ADV can occur before or after

v, but that V MUST occur in 2nd position. Note that this is 2nd position counting constituents, not words.

- Other kinds of variation can be obtained via bracketing. So for example {(N, V) CONJ (N, V)} means that the N's and V's can change order, but that the CONJ must come between each group. If we had {(N,V),CONJ,(N,V)} Then the N's and V's must occur next to each other, but each group doesn't not have to be separated by the CONJ, which can occur at the start, in the middle, or at the end, but which cannot break up an {N,V} group.

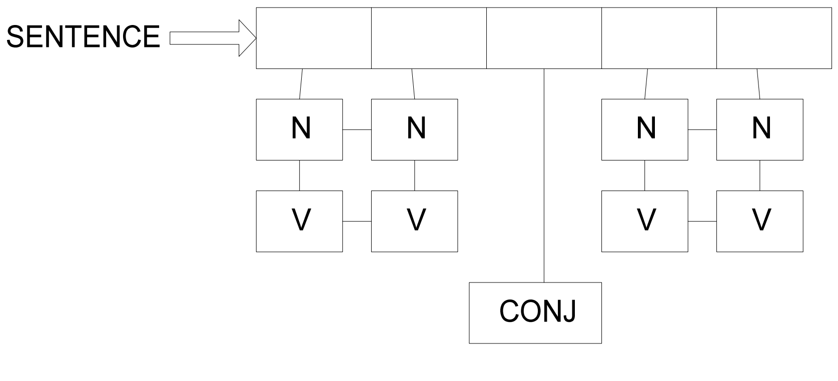
Modifications to the parsing algorithm.

Parsing was achieved via a structure that encoded all the possible orderings of a rule. So for example the rule CORE→N, V, N would become



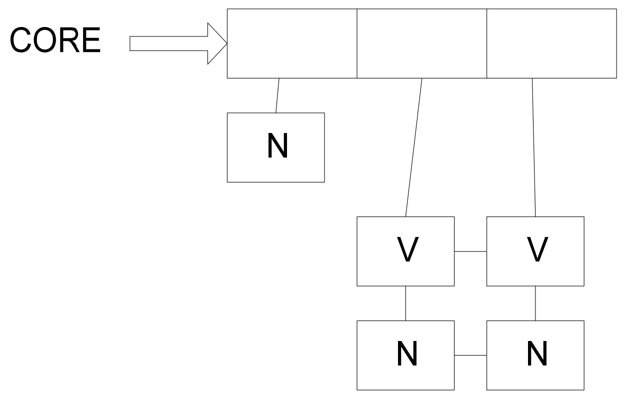
This means that N or V can occur in any position and N has to occur twice. The lines between the boxes enable the "rule" to be updated as elements are found.

Using this schema, SENTENCE→(N,V) CONJ (N,V) would become

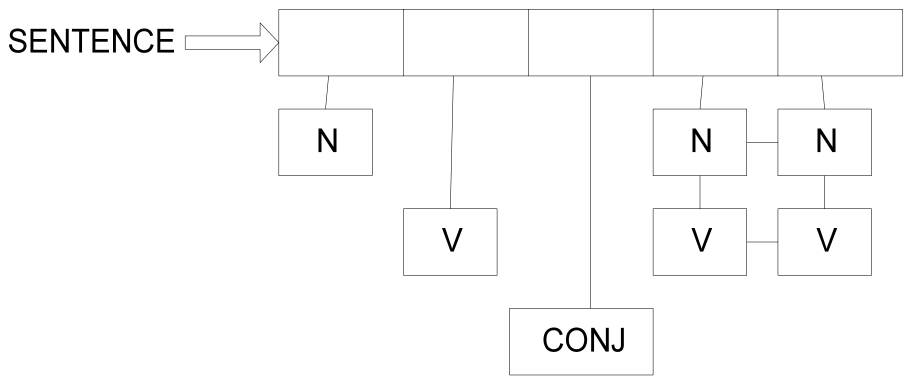


In this case, the CONJ in the middle is by itself because it has to occur in this position because the grouping word order is fixed. The groupings of N's and V's show where the free word ordering can occur.

To apply a rule, the first column of the left hand side of the rule is searched for the token. Any tokens that do not match are deleted along with the path that leads from them. In the first example, after an N is found, we would be left with



And in the second example, after an N is found we would be left with



Note that in order for the rule to be satisfied, we *must* find a V and then a CONJ: there are no options for position 2 once the element for position 1 has been established.

In this way, we can keep track of which elements of a rule have been found and which are still to be found. Changes in ordering with respect to the template are catered for by making sure that all instantiated rules point back to the appropriate leaves of the rule template, as described above.

The different possibilities for each rule are obtained via a breadth first search method that treats tokens in brackets as blocks. Then the problem becomes one of working out the number of ways that blocks of different sizes will fit into the number of slots in the rule.

Results

Preliminary results of applying these algorithms to student texts are very promising, but some issues have been highlighted. The method parses relatively simple sentences correctly and the main arguments and verbs are found. In addition, some very long and complicated sentences are parsed correctly and many kinds of grammatical errors do not cause any problems.

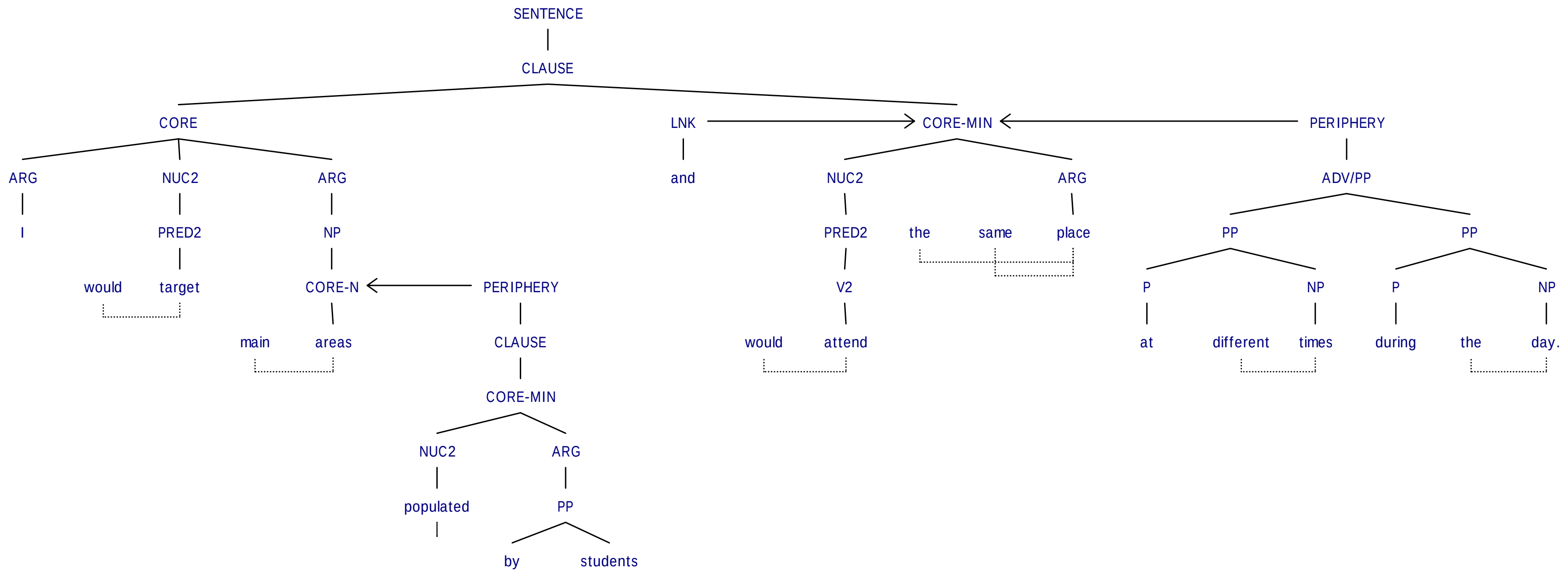


Figure 2: An example of a correctly parsed sentence.

An example of a correctly parsed sentence is “I would target main areas populated by students and would attend the same place at different times and during the day.” The parse tree for this example is given in figure 2. Note that the complex object “main areas populated by students” has been parsed correctly and that the tree attaches the qualifying phrase to “area” so that it is clear what is being qualified. An important source of ambiguity in English sentences is caused by prepositional phrases and this is a main cause of multiple parses of a sentence. In this example, the phrases “at different times” and “during the day” are placed together in the periphery of the CORE, although arguably they should have a different structure. This is a design decision to limit the number of parses. This kind of information needs semantic information to sort out what attaches to what. This cannot be obtained purely from the syntax.

An example of an ungrammatical sentence that is correctly parsed is “Results from the observations would be less bias if the sample again was not limit the students in the labs between 9:30 and 10:30 on a Thursday morning.” This sentence parses correctly because the affix that should be on “limit” is an operator and the correctness of the operators is not checked during the parsing process. The word “bias” is labelled as a noun and gets attached as the second argument to “would be”, although it should be “biased”, which would get it labelled as an adjective. Despite these errors, the meaning of the sentence is clear and the parse will enable the meaning to be deduced.

The sentence “Therefore, asking only the students present on a Thursday morning will exclude all the students that either have no lessons or are not present” produces two parses: once correct and one incorrect. The incorrect parse breaks up “Thursday morning” to give two clauses:

a. Asking only students present on a Thursday
b. Morning will exclude all the students that either have no lessons or are not present

In the first clause, the subject is “asking only students”, the main verb is “present” and the object is “on a Thursday morning”. This does not make sense, but it is syntactically correct as far as the main constituents are concerned. Similarly, the second clause is also syntactically correct, although it does not make sense. There are two ways of eliminating this parse. The first is to do a semantic analysis; the second is to not allow two clauses juxtaposed next to each other without punctuation such as a comma. However, students tend to not be very good at getting their punctuation correct. The current implementation of the parsing algorithm ignores all punctuation other than full stops for this reason.

An issue that makes parsing problematic is that of adverbs. These tend to be allowed to occur within several places within the core and some, such as yesterday, modify groups of words rather than a single word. The best solution, given their relative freedom of placing and the fact that sorting out where best to put them is more a meaning than a syntactic issue, would be to remove them and work out where they belong once the main verb and arguments have been identified.

Most of the above issues have to be left to an analysis of meaning to sort out the correct parse. There is no clear division between syntax and semantics. However there is another issue that has been highlighted to do with grammar and punctuation. How tolerant of errors should the system be? We have shown that errors in the operators do not cause problems for the parser, and errors in the placing of adverbs are relatively easy to deal with, but errors in the main constituents are not handled. For example the phrase “the main people you need to ask will not be in the labs so early unless that have got work to hand in” occurs in one of the texts. The current algorithm will not handle these kinds of mistakes. But should the system be able to handle these kinds of mistakes, or should students be encouraged to improve their writing skills?

Conclusion

We argue that this approach, though still under development, potentially has huge benefits for students and staff in higher education and could, with further improvements, form one building block in constructing a new paradigm for CAA. Our intention is to use this as the first stage in a system that uses a new semantic framework, ULM (Universal Lexical Metalanguage) (Guest and Mairal Usón, 2005), to compare the meaning of student texts with a (single) model answer.

References

Beevers, C E, Foster, M G, and McGuire, GR. 1989. Integrating Formative Evaluation into a Learner Centred Revision Course. *British Journal of Educational Technology*:115-119.

Brown, S, Race, P, and Bull, J. 1999. *Computer Assisted Learning in Higher Education*. London: Kogan Page.

Chung, Hoojung, and Rim, Hae-Chang. 2004. Unlexicalized Dependency Parser for Variable Word Order Languages based on Local Contextual Pattern. [Feb 15-21]. *Lecture Notes in Computer Science: Computational Linguistics and Intelligent Text Processing (5th International Conference CICLING)* 2945:112-123.

Covington, Michael A. 2003. A Free Word Order Dependency Parser in Prolog.

Guest, E, and Mairal Usón, Ricardo. 2005. Lexical Representation Based on a Universal Metalanguage. *RAEL, Revista Española de Lingüística Aplicada* 4:125-173.

Holan, Tomáš. 2002. Dependency Analyser Configurable by Measures. *Text, Speech and Dialogue 5th International Conference TSD*:81-88.

Hou, Lijun, and Cercone, Nick. 2001. Extracting Meaningful Semantic Information with EMATISE: an HPSG-Based Internet Search Engine Parser. *IEEE International Conference on Systems, Man, and Cybernetics* 5:2858-2866.

Kešelj, Valdo. 2001. Modular HPSG. *IEEE International Conference on Systems, Man, and Cybernetics* 5:2867-2872.

Landauer, Thomas K., Laham, Darrell, Rehder, Bob, and Schreiner, M. E. 1997. How well can Passage Meaning be Derived without using Word Order? A Comparison of Latent Semantic Analysis and Humans. *Proceedings of 19th Annual Conference of the Cognitive Science Society*:412-417.

Pérez, D, and Alfonsa, E. 2005. Adapting the Automatic Assessment of Free-Text Answers to the Students. Paper presented at *9th Computer Assisted Assessment Conference*, Loughborough, UK.

Rust, C. 2002. The Impact of Assessment on Student Learning. *Active Learning in Higher Education* 3:145-158.

Sadler, D R. 1989. Formative Assessment and the Design of Instructional Systems. *Instructional Science* 18:119-144.

Sadler, D R. 1998. Formative Assessment: Revisting the Territory. *Assessmnet in Education: Principles, Policy and Practice* 5.

Sambell, K, and Hubbard, A. 2004. The Role of Formative 'Low Stakes' Assessment in Supporting Non-Traditional Students' Retention and Progression in Higher Education: Student Perspectives. *Widening Participation adn Lifelong Learning* 6:25-36.

Sukkarieh, Jana Z, Pulman, Stephen G, and Raikes, Nicholas. 2003. Auto-marking: using computational linguistics to score short, free text responses. Paper presented at *International Association of Educational Assessment*, Manchester, UK.

Sukkarieh, Jana Z, Pulman, Stephen G, and Raikes, Nicholas. 2004. Auto-Marking 2: An Update on the UCLES-Oxford University research into using Computational Linguistics to Score Short, Free Text Responses. Paper presented at *International Association of Educational Assessment*, Philadephia.

Van Valin, Robert D Jr, and LaPolla, R. 1997. *Syntax: Structure, Meaning and Function*. Cambridge: Cambridge University Press.

Van Valin, Robert D Jr. 2005. *Exploring the Syntax-Semantics Interface*: Cambridge University Press.

Wahlster, Wolfgang. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*: Springer.

Wiemer-Hastings, Peter. 2001. Rules for Syntax, Vectors for Semantics. *Proceedings of 22nd Annual Conference of the Cognitive Science Society*.

Yorke, M. 2001. Formative Assessment and its Relevance to Retention. *Higher Education Research and Development* 20:115-126.