CAA OF SHORT NON-MCQ ANSWERS

David Callear, Jenny Jerrams-Smith and Victor Soh

CAA of Short Non-MCQ Answers

Dr. David Callear
Dr. Jenny Jerrams-Smith
Victor Soh

University of Portsmouth
Dept. of Info. Systems
Burnaby Terrace 1.17
1-8 Burnaby Road
Portsmouth
Hants PO1 3AE
United Kingdom

DavidCallear@port.ac.uk
Jenny.Jerrams-Smith@port.ac.uk
Victor.Soh@yahoo.com

Tel: 02392 84 6418
Fax: 02392 84 6402

Abstract

This paper presents a new approach for the computer-assisted assessment (CAA) of non- multiple choice questions (Non-MCQ) type and short answers given by students. The technique is developed for the assessment of text contents of free text answers to questions of factual disciplines.

The Automated Text Marker (ATM) prototype automatically breaks down an expertly written model answer, to a closed-ended question, into the smallest viable unit of concepts with their dependencies accounted for by automatically tagging the resultant concepts and their dependencies with numbers. The same process is applied to each student's answer and the resultant concepts and their dependencies are then pattern-matched with those of the model examiner's answer.

Two main components of ATM are the syntax and semantics analysers. In a prototype test, ATM provides for one score for the grammars and the other for the text contents.
The focus of this paper is on semantic analysis of text contents since the syntactic analysis of sentences has been generally and successfully automated.

Various examples of sentences of different factual disciplines such as those of Prolog programming, psychology and biology-related fields are analysed. Justifications for these analyses of sentences are provided and the corresponding prototype tests are conducted. The expected results from prototyping using ATM are obtained, indicating the reliability and feasibility of this new approach for the detailed assessment of text contents incorporating word order.

Work is currently underway for building a larger and more comprehensive ATM system for analysing and assessing text components larger than sentences such as paragraphs and whole text passages. Unlike existing computerised assessment systems, ATM is not a *predictive* system, although, like a human assessor, it is not perfect.

Keywords

Computer-Assisted Assessment, Intelligent and Expert Systems, Natural Language Processing, Prolog, Syntax, Semantics, Structured Knowledge Representation Schemes, Clustering and Management of Concept Representations.

Introduction

A new approach for computer-assisted assessment (CAA) of free text answers given by students is applied to assess the short answer format.

The assessment of students by multiple choice questions (MCQ) has been criticised for not measuring higher order cognitive skills and thus, not authentic. On the other hand, it is difficult for an instructor to assess students uniformly in a class of 200 students when pressured for time, under ordinary circumstances. A computerised assessment system would at least provide a 'double marking' and alert the instructor when there are large discrepancies between the grades he or she has given when compared to those produced by the computerised systems.

Existing computerised systems (Burstein et al, 1998a; Educational Testing Service, 1998; Burstein et al, 1998b; Dessus et al, 2000; Foltz et al, 1999; Landauer and Dumais, 1997; Landauer, Foltz and Laham, 1998; Landauer et al, 1997; Latent Semantic Analysis, 1997; Page, 1966, 1968, 1994) do not assess text contents at formative level and therefore, are not suitable for the short answer format. Moreover, they are predictive systems, although some of them assess text contents by matching weighted keywords only (Burstein et al, 1998b; Latent Semantic Analysis, 1997). These systems are not useful for assessing factual disciplines which call for explicit, short and concise answers. The word order is not taken into account. A summary of these various systems is provided in a paper entitled, *Approaches to The Computerised Assessment of Free Text Responses* (Whittington and Hunt, 1999).

The authors' *Automated Text Marker (ATM)* is developed for assessing text contents and is particularly suitable for assessing short answers to closed-ended questions of factual disciplines. An examiner's model answer is automatically segmented into smaller concepts with their dependencies accounted for, whenever necessary. The same process is applied to a student's answer which is then pattern-matched with the model answer and assessed. ATM assesses basic grammars and text contents. The word order is taken into account.

The scope of this paper consists of an introduction to ATM, the methodology and formalism used, as well as analyses of various short answers by the system and their respective justifications.

ATM Basic Architecture

The basic architecture of ATM is shown in Figure 1. ATM is written in Prolog which is very good for prototyping. Each part of a Prolog program can be automatically tested. Prolog is particularly adept at handling words and sentences, by treating them as lists.

One of the two main components of ATM is the syntax analyser. A simple syntax analyser is shown in Appendix A. The program source codes shown are quite readable and explicit. It is used to check the grammar of each input sentence. How complex is a sentence allowed is thus, constrained by the syntax analyser. An option for a user to finally submit an answer, irrespective of whether it is grammatically correct or not or the sentence is excessively convoluted and ambiguous or otherwise, is provided in the interactive ATM syntax analyser. The grammar can be augmented to include a wide-coverage, context-free and formalised grammatical description such as the *Generalised Phrase Structure Grammar (GPSG)* (Gazdar et al, 1985; Bennett, 1995). The large scale *Alvey Natural Language Tools (ANLT)* (Taylor, 1996-1998) are also built around the GPSG formalism.

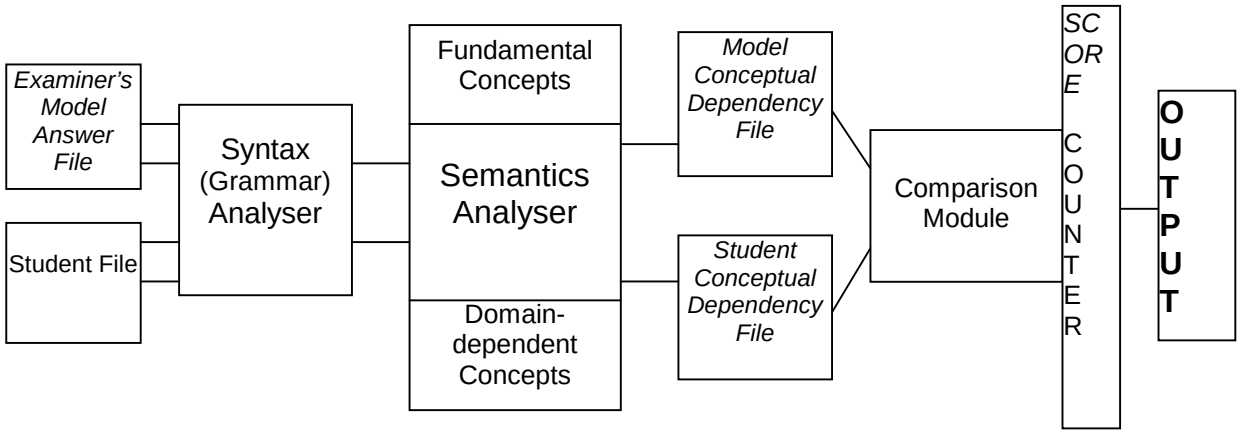

Figure 1. ATM Basic Architecture

The other main component of ATM is the semantics analyser which is the **focus** of this paper, since syntactic analysis of sentences has been successfully automated. In order to model the deep semantic structures of natural language, this semantics analyser is separately implemented. A context-sensitive grammar would drastically increase the program complexity and is not sufficient to model the deep semantic and pragmatic representations of the meanings of sentences, assuming that the intended meanings reside explicitly in the text.

ATM provides for two separate scores, one for the grammar and one for the text content. It is up to the examiner or course instructor to assign the weight for each score when calculating the overall or final score for each student.

Analyses of Short and Concise Sentences Using ATM Prototype

Research in natural language processing (NLP) has spanned a period of more than 40 years and the field has never fully matured. The development of the Prolog programming language is synonymous with the research in NLP. Insights gained from the NLP research can now be exploited and allowed to move from applications in toy domains to suitably constrained, realistic and practical applications such as the *CAA of Short Non-MCQ Answers*, the *Computer-Assisted Language Learning (CALL)* systems, the application of the NLP techniques in information retrieval or digital libraries, etc.

Various variants of a sentence with a *relative* level of *detail* are shown in Figures 2, 3 and

An infection is the invasion and multiplication of microorganisms in body tissue that produce signs and symptoms as well as an immunologic response.

Figure 2. Q: What is an infection? Variant answer A

An infection in body tissue is the invasion and multiplication of microorganisms that produce signs and symptoms as well as an immunologic response.

Figure 3. Q: What is an infection? Variant answer B

The invasion and multiplication of microorganisms in body tissue that produce signs and symptoms as well as an immunologic response are an infection.

Figure 4. Q: What is an infection? Variant answer C

DEPENDENCY GROUP 1
group(1,([infection] ➔ [is] ➔ [invasion])).
group(1,([infection] ➔ [is] ➔ [multiplication])).
group(1,([invasion] ➔ [of] ➔ [microorganism])).
group(1,([multiplication] ➔ [of] ➔ [microorganism])).

DEPENDENCY GROUP 2
group(2,([group(1)] ➔ [in] ➔ [body tissue])).
group(2,([infection] ➔ [in] ➔ [body tissue])).

DEPENDENCY GROUP 3
group(3,([group(1)] ➔ [produce] ➔ [sign])).
group(3,([group(1)] ➔ [produce] ➔ [symptom])).
group(3,([group(1)] ➔ [produce] ➔ [immune response])).

DEPENDENCY GROUP 4
group(4,([group(2)] ➔ [produce] ➔ [sign])).
group(4,([group(2)] ➔ [produce] ➔ [symptom])).
group(4,([group(2)] ➔ [produce] ➔ [immune response])).

Maximum score : 12 points

Figure 5. Q: What is an infection? CD Form

4. These variants have identical meanings composed from identical root words in various combinations. These are the answers to the question *What is an infection?* Their conceptual dependency (CD) groups, at the *corresponding relative* level of *granularity*, are shown in Figure 5. For clarity, the CD groups are shown in an output form and are not in the same way as the data are represented in Prolog within the program. This is explained in a later section on **'ATM Structured Representation Schemes'**. Concepts linked by the verb, *is,* can be placed on either sides of the verb and are thus, interchangeable as in Figures 2 and 4. In contrast, the concepts on either sides of, for example *because*, are not interchangeable. Note that the maximum score for this example question is fixed at 12 points for twelve concepts which match.

An example sentence of Prolog programming is shown in Figure 6. The CD groups for the answer to the question *What is the relation between a fact and a rule?* is given in Figure 7. The last dependency group expresses the fact that the second concept is the converse of the first. The determiners have disappeared, and a mainstay of the efficient implementation of this method is a simple analysis initially which enables non-necessary words to be discarded at the start. Note that the maximum score for this model answer is fixed at 8 points for eight concepts which match.

A fact in Prolog is a rule with no body, and conversely a rule is a fact with a body.

Figure 6. Q: What is the relation between a fact and a rule?
Model Answer

DEPENDENCY GROUP 1
group(1,([fact] ➔ [is] ➔ [rule])).
group(1,([rule] ➔ [with] ➔ [body])).
group(1,([no] ➔ [describe] ➔ [body])).
group(1,([fact] ➔ [in] ➔ [prolog])).

DEPENDENCY GROUP 2
group(2,([rule] ➔ [is] ➔ [fact])).
group(2,([fact] ➔ [with] ➔ [body])).
group(2,([rule] ➔ [in] ➔ [prolog])).

DEPENDENCY GROUP 3
group(3,([group(1)] ➔ [converse] ➔ [group(2)])).

Maximum score : 8 points

Figure 7. Q: What is the relation between a fact and a rule? CD Form

ATM Methodology, Canonicality and Practical Simplifications

As shown in the preceding section, each fragment of concept is either *totally independent* (a dependency group by itself) or *falls under a major dependency group*, and is *automatically* given a numerical tag (*number*). Each numbered dependency group represents the *context within which* fragments of concept must be reclustered and segregated. These major dependency groups can be further related to each other so that *successively* larger dependency groups are generated and numbered automatically.

Further prototyping of ATM is done by using a *standard* set of *fundamental concepts* or *primitives* and *domain-dependent concepts,* with synonyms and metonyms, where appropriate, on a thesaurus approach. An example pair of sentences is shown in Figure 8. For any two (2) sentences having the same meaning, only one (1) representation is used. These two sentences are reduced to *canonical* form as shown in Figure 9, which is again in output form. Note that the maximum score here is fixed at 2 points for a total of two concepts/dependency groups which match.

Taxes are increased because of a budget deficit.

Due to shortage of funds, taxes have gone up.

Synonyms:

have gone up ⇔ increase

due to ⇔ because

shortage of fund ⇔ budget deficit

Figure 8. Example pair of sentences

DEPENDENCY GROUP 1
group(1,([tax] ➔ [is] ➔ [increase])).

DEPENDENCY GROUP 2
group(2,([group(1)] ➔ [because] ➔ [budget deficit])).

Maximum score : 2 points

Figure 9. CD form of example pair of sentences

Prolog has a built-in syntax for handling grammar by breaking sentences down into phrases. A few of the simplifications presently used are:

(a) Definite and indefinite articles, and other 'unnecessary' words, are discarded.

(b) Complex sentences are broken down into simple sentences.

(c) Subordinate clauses are converted to simple sentences.

(d) Simple sentences contain one verb, which is a dependency.

(e) Adverbs, adjectives, conjunctions and prepositions represent further dependencies.

(f) A dependency can be represented as a duad : dependency – concept.

(g) A dependency can be represented as a triad: concept – dependency – concept.

(h) Each sentence represents a group of dependencies.

ATM Structured Representation Schemes

The paradigm used in the ATM prototype enables the *separation* of knowledge representation into a *fundamental* level, and a *domain-dependent* level making use of *case frames*. The Prolog equivalents of these case frames are *structures* or *structure predicates* as follow :

Examples of fundamental concepts:

word, bird, college, orange, 'David', generalisation, noun, etc.

Examples of domain-dependent concepts:

```
verb(give(NUMBER, DONOR, RECIPIENT, OBJECT)).
verb(see(NUMBER, VARIABLE1, _, Etc.)).
adjective(arduous(NUMBER, TASK).
adverb(surprisingly(NUMBER,adjective(NUMBER, ADJECTIVE))).
conjunction(while(NUMBER ,VARIABLE1, VARIABLE2)).
preposition(in(NUMBER,pointer_or_dependency_group(NUMBER),
                'body tissue', Etc.)).
```

Note that verbs, adjectives, adverbs, prepositions, conjunctions, etc. are *structure-building* words as opposed to simple nouns. The *Sharing* and *reuse* of a representation are accomplished by automatically tagging each case frame with a *number* as shown in the above examples. The variables (arguments) of case frames (predicates) can be *pointers* to other case frames. The underscore, "_", refers to an *anonymous* variable in Prolog. During *run-time*, these variables of case frames are bound or *instantiated* through *unification* with the domain text corpora.

ATM *structured knowledge representation schemes* are *powerful* in terms of their *expressiveness* and *granularity* for the purpose of modelling a natural language.

Discussion

Recent research (Coniam, 1995; Willis,1992; COLLINS COBUILD, 1992) suggests that a lexicon of the most common 2,500 words of English accounts for 80% of all English text. COBUILD research (COLLINS COBUILD, 1992) has established that the most frequent 1500 and 700 words of English account for 76% and 70% respectively of all English text, and this is implementable in the larger and more comprehensive ATM system. The number of operation words to be implemented is estimated at less than 100. These operation words include prepositions, conjunctions, pronouns, common verbs, etc. but exclude nouns and uncommon verbs.

In order for the larger ATM system to work, highly convoluted sentences, which would be difficult even for a human assessor to unambiguously figure out the long-distance dependencies, would not accepted by the interactive ATM syntax analyser. Theoretically, a sentence can consist of, for example, 5000 words, but this will not be acceptable by academic standards.

The ATM system provides for a considerable enhancement in assessment as opposed to a keyword assessment. When combined with some techniques of the existing computerised assessment of free text systems, formidable systems are conceivable.

Conclusions

Feasibility studies indicate that the ATM prototype provides for an effective *conceptual pattern matching* of student's answers with those of model examiner's answers.

Text passages are automatically broken down into their *smallest viable unit of concepts*. Basic concepts and their dependencies are reclustered and implemented as case frames with *numerical tags* to identify from which *context* these concepts arise, and can thus be grouped together.

Successively larger concepts or dependency groups are derived through systematic replacement of the variables of appropriate case frames with pointers to other dependency groups or case frames.

This project provides the basis for solving the words-on-a-page problem, and the methodology has been shown to be promising through prototype tests.

Appendix A

Example of a Prolog Syntax Analyser

```
start:-write('Enter one complete sentence:'), nl, nl,
        getsent(Sentence),
        test(Sentence).

test(Sentence):-phrase(sentence, Sentence), write('Sentence is correct.'), nl.
test(_):-write('Sentence is incorrect.'), nl.

/* Basic Sentence */

sentence -->noun_phrase(N), verb_prep_phrase(N), fullstop.

/* Noun phrases. Accept phrases joined by a conjunction */

noun_phrase(plural) --> noun_group(_), conjunction(and), noun_group(_), !.
noun_phrase(N)-->noun_group(N).
```

```
/* Nouns  grouped with other words.  Accepts determiners and adjectives and
   distinguishes between nouns, proper nouns and pronouns */

noun_group(N)-->determiner(N), noun_adj_group(N), !.
noun_group(plural)-->noun_adj_group(plural), !.
noun_group(N)-->adjective, propernoun(N), !.
noun_group(N)-->propernoun(N), !.
noun_group(N)-->pronoun(N).

/* Deals with nouns  grouped with up to two adjectives */

noun_adj_group(N)-->adjective,adjective,noun(N), !.
noun_adj_group(N)-->adjective,noun(N), !.
noun_adj_group(N)-->noun(N).

/* Take care of prepositional phrases */

verb_prep_phrase(N)-->verb_phrase(N), prepositional_phrase,
                              prepositional_phrase, prepositional_phrase, !.
verb_prep_phrase(N)-->verb_phrase(N), prepositional_phrase,
prepositional_phrase, !.
verb_prep_phrase(N)-->verb_phrase(N), prepositional_phrase, !.
verb_prep_phrase(N)-->verb_phrase(N).

/* Verb phrases.  Deal with verbs grouped with up to two
    noun phrases, eg. He gave (her)(the book). */

verb_phrase(N)-->verb_group(N), noun_phrase(_), noun_phrase(_), !.
verb_phrase(N)-->verb_group(N), noun_phrase(_), !.
verb_phrase(N)-->verb_group(N).

/* These  cater for up to two  participles, eg.  He is going.  He is sitting reading. */

verb_group(N)-->verb(N), participle, participle, !.
verb_group(N)-->verb(N), participle, !.
verb_group(N)-->verb(N).

/* Prepositional phrases. Deals with up to two prepositions,
    eg. He goes to bed. He goes in to dinner. */

prepositional_phrase-->preposition, preposition, noun_phrase(_), !.
prepositional_phrase -->preposition, noun_phrase(_).
/* General vocabulary */

determiner(singular)-->[the].   determiner(singular)-->[their].   determiner(singular)--
>[a].
determiner(singular)-->[her].
```

```
determiner(plural)-->[the].  determiner(plural)-->[their].  determiner(plural)-->[some].

conjunction --> [and].  fullstop -->[.].

/* Some vocabulary */

propernoun(singular)--->[barbara].  propernoun(singular) --> [david].

pronoun(plural) --> [they].  pronoun(singular) --> [he].  pronoun(singular) --> [she].

verb(singular) -->[is].  verb(singular) -->[goes].  verb(plural) -->[are].  verb(plural) -->[go].

participle --> [sitting].  participle --> [going].  participle --> [reading].

preposition --> [in].  preposition --> [at].  preposition --> [on].  preposition --> [to].

adjective --> [useful].

noun(singular) --> [student].  noun(singular) --> [pupil].  noun(singular) --> [book].
noun(singular) --> [paper].  noun(singular) --> [journal].  noun(singular) --> [library].
noun(singular) --> [table].  noun(singular) --> [chair].  noun(singular) --> [bed].
noun(singular) --> [dinner].
noun(plural) --> [dinner].  noun(plural) --> [beds].  noun(plural) --> [pupils].
noun(plural) --> [students].  noun(plural) --> [books].  noun(plural) --> [papers].
noun(plural) --> [chairs].  noun(plural) --> [journals].  noun(plural) --> [tables].
noun(plural) --> [libraries].

/* Input rules */

getsent([W|Ws]):-get0(C), readword(C,W,C1), restsent(W,C1,Ws).
restsent(W,_,[]):-lastword(W), !.
restsent(W,C,[W1|Ws]):-readword(C,W1,C1), restsent(W1,C1,Ws).
readword(C,W,C1):-single_character(C), !,  name(W,[C]), get0(C1).
readword(C,W,C2):-in_word(C,NewC), !, get0(C1),
                  restword(C1,Cs,C2), name(W,[NewC|Cs]).
readword(C,W,C2):-get0(C1), readword(C1,W,C2).
restword(C,[NewC|Cs],C2):-in_word(C,NewC), !, get0(C1), restword(C1,Cs,C2).
restword(C,[],C).
single_character(44).  single_character(59).  single_character(58).
single_character(63).
single_character(33).  single_character(46).
in_word(C,C):-C>96,C<123.  in_word(C,C):-C>47,C<58.
in_word(39,39).  in_word(45,45).
```

```
in_word(C,L):-C>64, C<91, L is C+32.
lastword('.').      lastword('!').      lastword('?').

/* END */
```

References

Bennett, P. (1995) *A Course in Generalised Phrase Structure Grammar* London: UCL Press.

Burstein, J., Kukich, K., Wolff, S., Lu, Chi. and Chodorow, M. (1998a) *Computer Analysis of Essays.* NCME Symposium on Automated Scoring. USA.

Burstein, J., Kukich, K., Wolff, S., Lu, Chi., Chodorow, M., Braden-Harder, L. and Harris, Mary Dee. (1998b) *Automated Scoring Using a Hybrid Feature Identification Technique*. Proceedings of the 36th Annual Meeting of the Association of Computational Linguistics. Montreal, Canada.

Coniam, D. J. (1995) *Partial Parsing : Software for Marking Linguistic Boundaries in English Text*. Ph.D. Thesis, School of Linguistics, University of Birmingham. Birmingham, UK: University of Birmingham.

Dessus, P., Lemaire, B. and Vernier, A. (2000) *Free Text Assessment in a Virtual Campus.* Proceedings of the 3rd International Conference on Human System Learning. Paris: Europia, pp. 61-75.

Educational Testing Service (ETS) (1998), *Electronic Rater (e-rater)* < http://www.ets.org/research/erater.html/ >.

Foltz, P.W., Laham, D. and Landauer, T.K. (1999*) Automated Essay Scoring : Applications to Educational Technology.* Proceedings of ED-MEDIA '99 Conference, AACE, Charlottesville, USA.

Gazdar, G., E. Klein, G. Pullum and I. Sag (1985) *Generalised Phrase Structure Grammar*. Oxford: Blackwell.

Landauer, T.K. and Dumais, S.T. (1997) *A Solution to Plato's Problem : The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge*. Psychological Review, vol. 104, pp. 211-240.

Landauer, T.K., Foltz, P.W. and Laham, D. (1998) *Introduction to Latent Semantic Analysis* Discourse Processes, vol. 25, pp. 259-284.

Landauer, T.K., Laham, D., Rehder, B 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 the 19th Annual Conference of the Cognitive Science Society.

Latent Semantic Analysis (LSA). (1997) *Intelligent Essay Assessor (IEA)* <http://lsa.colorado.edu/ >.

Page, E.B. (1994) *New Computer Grading of Student Prose : Using Modern Concepts and Software*. Journal of Experimental Education, vol. 62, no. 2, pp. 127-142.

Page, E.B. (1966) *The Imminence of Grading Essays by Compute*r. Phi Delta Kappan, vol.47 (Jan), pp. 238-243.

Page, E.B. (1968) *The Use of the Computer in Analysing Student Essays*. International Review of Education, vol. 4, pp. 210-224.

Taylor, et al. (1996-1998) *Alvey Natural Language Tools (ANLT)* <http://info.ox.ac.uk/ctitext/resguide/resources/a145.html >.

Whittington, D. and Hunt, H. (1999) *Approaches to The Computerised Assessment of Free Text Responses* Proceedings of the 3rd. International Conference on Computer Assisted Assessment. Loughborough, UK.

Willis, D. (1992) *The Lexical Syllabus* COLLINS COBUILD. UK: Collins.