Auto-marking: using computational linguistics to score short, free text responses

Jana Z. Sukkarieh[1], Stephen G. Pulman[1] and Nicholas Raikes[2]

[1] Computational Linguistics Group, Centre for Linguistics and Philology, Walton Street, Oxford OX1 2HG, United Kingdom. Email: first.lastname@clg.ox.ac.uk

[2] Interactive Technologies in Assessment and Learning (ITAL) Unit, University of Cambridge Local Examinations Syndicate (UCLES[†]), 1 Hills Road, Cambridge CB1 2EU, United Kingdom. Email: N.Raikes@ucles-red.cam.ac.uk

Abstract

Many of UCLES' academic examinations make extensive use of questions that require candidates to write one or two sentences. For example, questions often ask candidates to state, to suggest, to describe, or to explain. These questions are a highly regarded and integral part of the examinations, and are also used extensively by teachers. A system that could partially or wholly automate valid marking of short, free text answers would therefore be valuable, but until

[†] The UCLES Group provides assessment services worldwide through three main business units.

- Cambridge-ESOL (English for speakers of other languages) provides examinations in English as a foreign language and qualifications for language teachers throughout the world.
- CIE (Cambridge International Examinations) provides international school examinations and international vocational awards.
- OCR (Oxford, Cambridge and RSA Examinations) provides general and vocational qualifications to schools, colleges, employers, and training providers in the UK.

For more information please visit http://www.ucles.org.uk

recently this has been thought impossible or impractical. Advances in computational linguistics, however, coupled with increasing penetration of computers into schools, have prompted several organisations to investigate automatic marking and its application to high or low stakes tests. One such organisation is UCLES, which is funding a three year study at Oxford University. Work began in summer 2002, and in this paper we describe the project and our progress to date using information extraction and retrieval techniques to mark General Certificate of Secondary Education (GCSE) biology answers. We also outline some of the uses that could be made of auto-marking, depending on its success.

Introduction

Traditionally, automatic marking (grading) has been restricted to item types such as multiple choice that narrowly constrain how students may respond. More open ended items have generally been considered unsuitable for machine marking because of the difficulty of coping with the myriad ways in which credit-worthy answers may be expressed. Successful automatic marking of free text answers would seem to presuppose an advanced level of performance in automated natural language understanding. However, recent advances in natural language processing (NLP) techniques have opened up the possibility of being able to automate the marking of free text responses typed into a computer without having to create systems that fully understand the answers. The project we introduce in the present paper was set up to investigate the application of NLP techniques to the marking of short, free text responses of up to around five lines.

Why short, free text responses?

Many of UCLES' academic examinations make heavy use of questions that require students to write one or two sentences and which are worth one or two marks. For example, questions often ask candidates to state, to suggest, to describe, or to explain, and even those papers that use multiple choice questions tend to use them sparingly and intersperse them with questions requiring a constructed response. These short answer questions are highly valued and integral to the examinations, and are also extensively used by teachers preparing students for the examinations. In these circumstances there will be little demand for an automatic marking system unless it can handle short, free text answers.

How might automatic marking be used?

High stakes assessment

Perhaps surprisingly, UCLES' least likely use for automatic marking in the near future is for marking high stakes examinations. There are two principal reasons for this. Firstly, automatic marking requires machine-readable answers, but most of UCLES' academic examinations are currently paper based – there was little reason to computerize them without a way of automatically marking short answers. It will take time to move from paper-based to computer based examinations. Secondly, the reliability, validity and defensibility of marking are of particular importance in high stakes examinations, and any automated system would have to pass a very comprehensive evaluation before it could be used to determine candidates' grades. It is doubtful whether any system likely to be available in the foreseeable future could completely eliminate human examiners for short answers. Perhaps the most likely scenario is that at first both automatic and human marking will be used. Under one possible initial system a student's "script" (i.e. answer file) might be split by question type, with the answers most readily susceptible to automatic marking (e.g. those for multiple choice questions) solely marked that way, and the more open ended answers marked both by human and computer, the automatic marks used as a check on the human marks and potentially grade-changing discrepancies resolved by further human markings. Over time, if trust in automatic marking increased, the range of answers solely marked automatically might be expanded, particularly if the automatic system provided a confidence rating for each short-answer mark it awarded. Human markers might then be primarily used when the uncertainty in a script's total mark, combined with the total's proximity to a grade cut score, combined to trigger a quality control procedure.

Low stakes assessment

By far the most likely first use of automatic marking, however, is for low stakes assessments designed to support teaching and learning.

At the simplest level, UCLES could provide schools with an automatic scoring service for online tests built from questions taken from past papers. The most straightforward system would just return marks, but marks on their own are of limited use to either students or teachers, since they contain no information about precisely what was wrong with a less than perfect answer. The service could be enhanced by also providing teachers with a transcript of their students' answers that prominently highlighted answers that were not given full marks. This transcript could be provided student by student, or question by question, and a teacher could use it to help identify any remedial action that was needed. In this way automatic marking might essentially be used by teachers as a "filter" that enabled them to reduce the time spent reading and ticking correct answers and to focus instead on the answers that really needed their expert attention.

More ambitiously, we could attempt to provide each student with automatic textual, formative feedback relevant to that student's answers. The practicality of doing this has still to be determined, but one approach might be as follows. Firstly – before automation – a panel of experienced teachers would be employed to look at a sample of student answers. If past-paper questions were used the sample answers could be taken from archived examination scripts. The teachers' task would be to sort each sample answer into a feedback category depending on its semantic content – their aim would be to use the smallest number of categories possible whilst ensuring that a category was specific enough to allow useful feedback to be written. Typically there would probably be far fewer feedback categories than answers, since many answers would only be superficially different and a small number of mistakes or misconceptions would be common to many answers awarded less than full marks. Next, the teachers would write the formative feedback for each category. In this way the system would be primed with a database of sample answers matched to appropriate formative feedback. When the system went "live", an answer submitted for marking would be compared automatically with each of the sample answers. The nearest matching answers would be identified, and if the matches were close enough to exceed some pre-determined threshold the formative feedback returned for the submitted answer would be that pre-written for the category into which the majority of these answers fell. If no sample answers were close enough the system would return a message indicating that it was unable to mark the answer submitted. The frequency of "non marks" and the actual relevance of feedback returned would need to be evaluated, but this method might enable useful feedback to be given automatically to many students. Perhaps the most useful additional feedback for teachers that this method might enable – in addition to the marks and edited transcripts mentioned previously – would be summary information for each question indicating the distribution of their students' answers amongst the different categories.

Previous work: automatic marking of essays

UCLES' interest in automatic marking of free text was stimulated by the development of systems capable of marking essays automatically. The most prominent of these are now described briefly.

Latent Semantic Analysis

The Intelligent Essay Assessor (IEA) was developed by Knowledge Analysis Technologies (KAT), Colorado (Foltz, Laham and Landauer, 2003). It uses Latent Semantic Analysis for scoring answers of students as part of a tutoring domain. Latent Semantic Analysis is based on word-document co-occurrence statistics in the training corpus represented as a matrix, which is subsequently decomposed, and then subjected to a dimensionality reduction technique. LSA is used to compare students' answers to model answers by calculating the distance between their corresponding vector projections (Graesser *et al.*, 2000). IEA has been tested in different ways, namely, comparing essays to ones that have been previously graded, to an ideal essay or *gold standard* (Wolfe *et al.*, 1998), to portions of the original text, or to sub-components of texts or essays (Foltz, 1996; Foltz, Britt and Perfetti, 1996). In blind testing, agreement with human examiners is high, between 85 and 91 percent.

The LSA technique evaluates content via the choice of words and does not take into account any syntactic information — it is a 'bag-of-words' approach and can be fooled. "It has no way of

knowing the difference between *The Germans bombed the British cities* and *The British bombed the German cities*" (Charles Perfetti)[‡]. It cannot deal with any of the favourite list of difficult phenomena for NLP systems, like negation, attachment, binding, predication, modification, scope ambiguities and so on. Researchers have tried to improve the performance of LSA by adding some syntactic and semantic information; for example, adding a part-of-speech (POS) to the given word (Wiemar-Hastings and Zipitria, 2001) or adding a part-of-speech to the previous word (Kanejiya, Kumar and Prasad, 2003). The results do not show a significant improvement over the basic technique.

A Hybrid Approach

E-rater[§] (or Essay-rater) is a system developed by the Education Testing Service (ETS) (Burstein *et al.*, 1998a; Burstein, Leacock and Swartz, 2001; Burstein *et al.*, 1998b). It has been used to rate GMAT (a business school admission test) and TWE essays (Test of Written English) for prospective university students. The system uses shallow parsing techniques to identify syntactic and discourse features. Content is checked by vectors of weighted content words. An essay that stays on the topic, is coherent as evidenced by use of discourse structures, and has a good vocabulary and varied syntactic structure is to have a higher grade. E-rater uses both NLP and statistical tools to model the decision of a human marker, and achieves impressive agreement with human markers when tested on unseen data (84–91%).

Another hybrid approach is described by Rosé *et al.* (2003) at the University of Pittsburgh. Their system evaluates qualitative physics questions using a hybrid approach between machine learning classification methods using features extracted from a linguistic analysis of a text and a naive Bayesian classification.

Automatic marking of short textual answers: information extraction

In the UK, Intelligent Assessment Technologies have developed an automatic short answer assessor called Automark[**] (Mitchell *et al.*, 2002). The system uses information extraction techniques in the sense that the content of a correct answer is specified in the form of a number of mark scheme templates. The text to be marked is fed into a parser (they use the Link Grammar parser (Sleator and Temperley, 1991; Sleator and Temperley, 1993)) and the parsed text is then compared to the already-defined templates or mark scheme. Mitchell *et al.* (2002) claim about 95% agreement with human markers in blind testing.

Callear, Jerrams-Smith and Soh (2001) at the University of Portsmouth also use pattern-matching techniques to mark short answers in programming languages, psychology and biology-related fields.

The UCLES application for automatic marking of short textual answers

At the time we began this project, we were not aware of the Intelligent Assessment Technologies work. However, we had already decided on the basis of reading about E-rater that information extraction techniques were a likely candidate for our application, since they do not require complete and accurate parsing, they are relatively robust in the face of ungrammatical and

[‡] *Teachers of Tomorrow?* http://www.wired.com/news/technology/0,1282,16009,00.html

[§] http://www.ets.org/research/erater.html

[**] You can find demos at http://www.intelligentassessment.com/demonstration.htm for English Comprehension and at http://examonline1.nsl.co.uk/ExamOnline/Jsp/index.jsp for Key Stage 2 Science National Test Questions.

incomplete sentences (of which GCSE scripts provide a plethora of examples), and they are fairly easy to implement quickly.

After an initial look at a sample of different GCSE and A-level examination papers we decided to begin with GCSE Biology exams where the answers are restricted to about 5 lines and deal with facts rather than subjective opinions or interpretation. A sample of these scripts was transcribed from the original hand-written versions into machine readable text. About 25% of the sample was retained by UCLES to use as unseen test data for our system.

Here are some example questions along with their answer keys — short, often very terse, descriptions of acceptable answers provided by examiners.

1 Write down two things about asexual reproduction in plants which is different from sexual reproduction.

Key:

Can be done at any time

Needs no flowers

Does not need 2 gametes/parents

No fertilisation

No meiosis involved

No genetic variation/clones/identical/same as parent plant

2 Explain what causes two twins to be identical.

Key:

Formed from the same bundle of cells/

same fertilised egg/same embryo /

formed from one egg and sperm /

mitosis forms identical cells

so genetic information the same/

same genes/same DNA/same chromosomes

3 Where could you detect the pulse and what causes it?

Key:

Found at wrist/temple/neck /

where an artery close to the skin can be pressed against

a bone or ankle for infants;

heart beating / blood surging in an artery /

wave down artery wall

Our starting point was six questions for which we had approximately 201 marked student answers per question to use for training, and approximately 60 answers per question that were held back for testing. Each answer was associated with the score given to it by expert examiners, and this

score was either 0 (incorrect), 1 (partially correct or incomplete) or 2 (correct and complete). We used a Hidden Markov Model part-of-speech tagger trained on the Penn Treebank corpus, and a Noun Phrase (NP) and Verb Group finite state machine (FSM) chunker to provide the input to the information extraction pattern matching phase. The NP network was induced from the Penn Treebank, and then tuned by hand. The Verb Group FSM (i.e. the Hallidayean constituent consisting of the verbal cluster without its complements) was written by hand.

Customisation and Shallow Processing

We have assessed the performance of the tagger on the data (students' answers). The following table gives an idea, in figures, about its performance on the training set we started with. A major source of inaccuracy for taggers is the presence of unknown words. The Wall Street Journal section of the Penn Treebank is not particularly rich in biological vocabulary, and so we would expect problems in this respect, even though the tagger includes some heuristics for guessing at unknown words. To factor out the unknown word issue we first ran the tagger on sentences which contained no words unknown to it (although the entry for the word might still not be the correct one, of course). Then we tested the tagger on a large random sample of answers, and finally on that same sample with all unknown words added (and spelling errors corrected).

Students' answers	Performance of the tagger
No Unknown Words in answers	Total # of words 4030 # of words tagged wrongly: 60
Random Answers	Total # of words: 14196 # of words tagged wrongly: 218
After correcting spelling errors (and adding all new words)	Total # of words: 14196 # of words tagged wrongly: 98

Here is a sample of the output of the tagger and chunker:

```
When/WRB [the/DT caterpillars/NNS]/NP [are/VBP feeding/VBG]/VG
on/IN [the/DT tomato/JJ plants/NNS]/NP,/, [a/DT chemical/NN]/NP
[is/VBZ released/VBN]/VG from/IN [the/DT plants/NNS]/NP./.
[This/DT chemical/NN]/NP [attracts/VBZ]/VG [the/DT wasps/NNS]/NP
[which/WDT]/NP [lay/VBP]/VG [eggs/NNS]/NP inside/IN
[the/DT caterpillars/NNS]/NP./. ,
```

where *WRB, DT, NNS, VBP, VBG, IN, JJ, VBZ, VBN* are, respectively, tags for adverbs that start with "wh", a determiner, plural noun, non-3rd person singular present, ing-verb, preposition, adjective, 3rd person singular present verb, gerund and present participle verb. *NP* and *VG* mark a noun phrase and a verb group respectively.

The Pattern-Matcher

Information extraction can only be used if a fairly determinate task specification and a clear criterion for success are given (Appelt and Israel, 1999). Our task is fairly determinate, namely, identify a right GCSE Biology answer, and it has a reasonably well-defined criterion for success. Information extraction consists of applying a set of patterns and templates to discover members of a fixed list of *named entities* and relations within the texts, occurring in a specific configuration.

In our first attempt, given a question and answer, we try to identify a chunk of text in the answer that qualifies for a mark. We do this at a fairly concrete level on the basis of particular collections

of keywords. In a more refined version, we would try to identify more abstract semantic elements, like relations, properties, etc. but we wanted to get a baseline system up and running as quickly as possible.

Information extraction patterns, the things that get from each answer the information relevant to the particular task, can be either discovered by a human or can be learned with machine learning algorithms, although to date these machine learning techniques have not reached human levels of accuracy in the building of information extraction systems. In this version of the system, we opted for the first way, namely, the knowledge-engineering approach. This also means that the grammar describing the patterns is constructed by hand and only someone who is familiar with the grammar and the system can modify the rules. Moreover, this approach requires a lot of labour, as will become evident below.

Patterns and Grammar

The 3 crucial steps in which to write extraction rules by hand can be found, among other references on information extraction, in Appelt and Israel (1999). These, in order, are:

1. Determine all the ways in which the target information is expressed in a given corpus.
2. Think of all the plausible variants of these ways.
3. Write appropriate patterns for those ways.

Clearly the intuition of the linguistic/knowledge engineer plays an important role. For each domain, this requires some training as one is looking for a tightly defined, mostly unambiguous set of patterns that cover precisely the ways the target information is expressed, yet written in a way that captures the linguistic generalisations that would make it unnecessary to enumerate all the possible ways of expressing it. For the biology task we abstracted the patterns over 3 sets of data. First, we fleshed out the compact key answers provided by the examiners. Second, we used our own version of the answers (we sat the exam ourselves and with the help of a recommended GCSE biology book (Jones and Jones, 2001) we provided answers for the questions). The last set of data we abstracted patterns over was the training data that UCLES provided for us. After checking out the variant ways answers could be written, we devised a simple language in which to write the patterns.

```
Pattern          ->    Word | Word/Cat | Symbol
                       | Variable | Disjunction
                       | Sequence | k(N, Sequence)
                       (N is upper limit of length of Sequence)
                       | k(N, Sequence, Pattern)
                       (Sequence NOT containing Pattern)
Disjunction      ->    {Pattern, ..., Pattern}
Sequence         ->    [Pattern, ..., Pattern]
Word             ->    sequence of characters
Cat              ->    NN | VB | VG ...
Symbol           ->    & | % | $ ...
Variable         ->    X | Y | Z ...
```

It is easy then to build up named macros expressing more complex concepts, such as ‘negated verb group’, or ‘NP headed by word *protein*’.

The following answers exist in the training data as true answers for Question 2, namely, *Explain what causes two twins to be identical*.

```
the egg after fertilisation splits in two
```

```
the egg was fertilised it split in two

one egg fertilised which split into two

the fertilised egg has divided into two

1 fertilised egg splits into two
```

These all imply *It is the same fertilised egg/embryo*, and variants of what is written above could be captured by a pattern like:

```
singular_det + <fertilised egg> +
  {<split>; <divide>; <break>} + {in, into} + <two_halves>

singular_det     = {the, one, 1, a, an}
<fertilised egg> = NP with the content of 'fertilised egg'
<split>          = {split, splits, splitting, has split, etc.}
<divide>         = {divides, which divide, has gone,
                    being broken...}
<two_halves>     = {two, 2, half, halves}
etc.
```

It is essential that the patterns use the linguistic knowledge we have at the moment, namely, the part-of-speech tags, the noun phrases and verb groups. In our previous example, the requirement that <fertilised egg> is an NP will exclude something like '*one sperm has fertilized more than one egg'* but accept something like '*an egg which is fertilized...*'.

Another difficulty we faced when writing patterns was that examiners allowed unexpected ways for students to convey a particular scientific concept. Consider the word *fertilisation*, where examiners seemed to accept *the sperm and the egg meet*, *when a sperm meets an egg* or *the sperm reached the egg* instead.

As we said earlier, intuition plays an important role in writing patterns and it needs training for a particular domain. The development cycle will be familiar to anyone with experience of information extraction applications:

Write some patterns.

Repeat until satisfied with results:

Run the system over a training corpus.

Examine output.

See where patterns over-generate, under-generate, etc.

Modify or add patterns...

The 'system' referred to in 'Run the system over ...' in the iterative process above was a Prolog meta-interpreter that searches for the patterns in a particular given answer.

A tool to help with pattern writing

Patterns have to be expressed precisely and in the correct language. In order to help content-experts construct patterns we have built a prototype application[††] which prompts the user to fill in paraphrases or other acceptable alternatives for key answers. These are then automatically translated into corresponding patterns suitable for the automatic marker. If a database of examiner-marked sample answers is available the system also lets the user try out the patterns against the sample answers. The results returned enable the user to identify cases where the automatic and examiner marks do not match, so that the list of alternative answers may be

[††] Robert Chilvers, our summer student, implemented the customisation tool.

refined. We hope that in this way the development cycle above may be implemented by content-experts without detailed knowledge of how the system works.

The Basic Marking Algorithm

With each question, we associated a set of patterns or rules. The set of rules for a particular question was then divided into bags or equivalence classes where the equivalence relation, *R*, is *convey the same message/info as*. Equivalence classes are represented by one of their members. *X* belongs to a class [**Rep_of_Class**][‡‡] if *X* bears *R* to Rep_of_Class. For example, 'only one parent' *R* 'just one parent'; [**only one parent**] = {*Pattern* | *Pattern R* 'only one parent'}. For Question 1, we have 6 equivalence classes, namely, [**only one parent**], [**clone**], [**no need to flower**], [**can be done at any time**], [**no fertilisation**], [**no meiosis involved**].

The key-answers given by the examiners determine the number of equivalence classes we have. Each equivalence class corresponds to 1 mark the examiners give. Assume we have *N* classes for a particular question:

Class 1 with {*C*11, *C*12, ..., $C2t_1$}
Class 2 with {*C*21, *C*22, ..., $C2t_2$}
...
Class *N* with {CN_1, CN_2, ..., CNt_n}

```
Given an answer A,
Repeat until no more rules/classes are available
   If A match Cik
      Then   Mark-till-Now is Mark-till-Now + 1
                If    Mark-till-Now = Full Mark
                      Then Exit
          %(ignoring Ci+1, ..., Cn i.e. the rest of the classes)
                      Else Ignore Cij for  k <  j  <=  ti
          %(i.e. ignore the rest of the rules in the same bag)
                           See if A matches any rule in Ci+1
          %(i.e. jump to the next Class and repeat process)
```

The procedure *match* takes the patterns as described by the grammar, case by case and handles them accordingly.

Some Anticipated Problems

Information extraction and shallow processing are not full natural language processing methods and so there will be many cases that we handle incorrectly:

- **The need for reasoning and making inferences:** Assume a student answers Question 1, above, with, *we do not have to wait until Spring*. An assessor that fails to infer *it can be done at any time* from the student's answer will give it a 0. Similarly, an answer like *don't have sperm or egg* will get a 0 if there is no mechanism to infer *no fertilisation*.
- **Students tend to use a negation of a negation (for an affirmative):** An answer like *won't be done only at a specific time* is the same as *will be done at any time*. An answer like *it is not formed from more than one egg and sperm* for Question 2, is the same as saying *formed from one egg and sperm*. This category is merely an instance of the need for more general reasoning and inference outlined above. We have given this case a

[‡‡] Representational note: we will always write the representative of a class in bold so that there is no confusion between an alternative of a pattern and a class of a pattern.

separate category because here, the wording of the answer is not very different, while in the general case, the wording can be completely different.

- **Contradictory or inconsistent information:** Other than logical contradiction like *needs fertilisation and does not need fertilisation*, an answer for Question 2 like *identical twins have the same chromosomes but different DNA* holds inconsistent scientific information that needs to be detected.

Some of these issues are reconfirmed in the students' actual answers.

In the next section, we report the results of the pattern-matcher on both the training data and the testing data, followed by a brief discussion.

Results

There were approximately 201 answers used as training data and 65 testing answers available for each question. The results are summarised in the following table:

	Training Data	Testing Data
Hits	88 %	88 %

'Hits' occur where the system's and the examiners' marks match (the percentage includes answers with mark 0). The results are for the first version of the system, and we find them highly encouraging. One would expect the system to be reasonably accurate in the training data, but to find no deterioration when moving to unseen data is very gratifying. This must mean that there is very little variation in the range of answers, and in particular that a training set of the size we have will very likely be enough (we have already tried with one third of the training data for 3 more questions and the percentage of hits is similar to the one mentioned in the table above).

Given that this was the first version of the system complete enough to test it is clear that we could get some further improvement by putting more work in on the patterns. However, there are two factors relevant here. Firstly, there is the amount of work involved in writing these patterns. It would be nice to be able to automate the task of customising the system to new questions. Secondly, there are several observations about the behaviour of unintelligent pattern matching of this sort which suggest that the cost/benefit ratio may become unfavourable after a short time. Recall that these patterns are not doing full natural language understanding. This means that there will always be a trade-off between high precision (recognising patterns accurately) and high recall (recognising all variants of patterns correctly). It is not guaranteed that for a particular application the right trade-off can be found. This suggests that it is worth experimenting with machine learning techniques in order to help with the process of customisation. If this cannot be achieved fully automatically we could at least investigate what help could be given to the developers via such techniques. For this reason we next turned to a simple machine learning method in order to develop a suitable experimental framework.

Nearest Neighbour Classification

We have already mentioned some suitable techniques. Latent Semantic Analysis is, from one point of view, just a way of classifying texts. Other researchers (Rosé *et al.* (2003) and Larkey (1998)) have also used text-classification techniques in grading essays in subject matters like physics, or law questions where a legal argument is to be expected in the text. While the accuracy of such systems may not be able to exceed that of hand-crafted systems (although this is not a proven fact), they nevertheless have the advantage of being automatically customisable to new domains needing no other expert knowledge than that of a human examiner.

In general all variants of these techniques begin with a set of examples with a known analysis (preferably covering the entire range of types of analysis). The size of the set can be as few as 100, although the larger the better. When this training phase is complete, new examples to be analysed are matched with old ones, or a combination of them, and the closest match determines the appropriate response. In our application, the 'analyses' are scores, and we assign the score of the nearest matching example to the input to be rated. The matching process may be quite complicated — we will have to experiment with different variations. The attraction of such a setup is that customising it to a new exam would be a matter of marking a few hundred scripts by hand to provide the training examples, once the appropriate matching and analysis schemes have been discovered.

We have begun by using almost the simplest posible text-classification method, known as the *k nearest neighbour* (KNN) technique (Mitchell, 1997). We decompose examples (or new input) into a set of *features*. These can be words, tuples of words, grammatical relations, synonym sets, combinations of these or whatever the linguistic analysis mechanism is capable of finding accurately. In our case, we began with word tokens, thus approximating a crude marking technique of spotting keywords in answers. We discard determiners and a few other function words which have very low discriminatory power, and assign to each content word which appears in the training set a weight, the so-called 'tf-idf' measure. This is *term frequency* (the number of times the term or feature appears in the example) multiplied by *inverse document frequency*, i.e. 1/(the number of times term or feature appears in all examples). Terms which do not distinguish well among examples will carry less weight. It is easy to see how to adapt such a measure to give higher weights to words that are associated with (in)correct answers, and less weight to words that occur in almost all answers.

Each example, and any new input, can be represented as a vector of weighted feature values, ordered and labelled in some canonical way. We can then calculate a cosine or similar distance measure between the training examples and the input to be scored.

To summarise:

1. Collect a set of training examples representing the main possible outcomes.
2. Represent each training example as a a vector of features, i.e. linguistic properties believed relevant. In the simplest case this will be keywords.

For example, we might categorise a set of answers on the basis of whether or not they contain one of the following key words:

	egg	fertilized	split	two	male	female	sperm
Answer1	1	1	0	0	1	0	1
Answer2	1	1	1	1	0	0	0
Answer3	1	0	1	0	0	0	1

In our case the vector values are numbers between 0 and 1, since they are weighted. For each training example, the score assigned by the examiners is known.

Classifying unseen answers

To classify a new answer we represent it as a vector and find which of the training examples it is nearest to, where the distance measure between two vectors x_n and y_n is defined as:

$$\mathrm{d}(x_n, y_n) = \log\left(\sum_{i=1}^{n} (x_i - y_i)^2\right)$$

This method can be generalised to find the *k* nearest neighbours, and then the most likely score will be that which occurs most frequently among the *k* neighbours.

Comparison and Discussion of Results

This is an extremely simple classification technique and we would not expect it to work very well. Like all 'bag-of-words' approaches, it completely ignores any higher level linguistic structure and so would represent *wasps lay eggs inside caterpillars* as the same answer as *caterpillars lay eggs inside wasps*, or indeed *eggs lay caterpillars wasps inside*. Nevertheless, it is still doing some work, as the following table shows. The figures in the table are the percentages of "hits", i.e. the number of times an automatic mark matched the human examiner's mark. The first row gives a naive *unigram* baseline score, computed by giving each answer the mark that occurred most frequently in the training set for the relevant question. The second row gives the *k nearest neighbour* results – 'k=3, +w, +f' means that we used 3 nearest neighbour matching, function words were filtered out and remaining words were weighted by inverted document frequency. Finally, the last row gives the information extraction score for comparison. Note that the information extraction approach was only implemented for 3 out of the 6 test questions, whereas it is no extra effort to do all the questions with the automatic techniques.

Question:	12(c)(i)	12(c)(ii)	13(b)(ii)	4(a)	5(a)(ii)	9(c)	Overall
Baseline	56 %	54 %	71 %	67 %	78 %	39 %	60 %
k=3, +w, +f	75 %	59 %	71 %	72 %	67 %	56 %	67 %
patterns	n/a	n/a	n/a	94 %	80 %	89 %	88 %

Clearly the pattern-matching method is doing better on the examples it dealt with. This does not mean, however, that the results would be like this for any choice of the feature set. It is possible that if we had more linguistic information represented in the vectors then the results of the KNN technique would improve.

Conclusions

We have demonstrated that information extraction techniques can be successfully used in the task of marking GCSE biology scripts. We have also shown that a relatively naive text classification method can score better than a simple baseline grading technique. There are many refinements to both kinds of approach that can be made: our eventual aim is to try to approach the accuracy of the information extraction method but using completely automatic machine learning techniques.

References

Appelt D. and Israel D. Introduction to information extraction technology. IJCAI 99 Tutorial.

Burstein J., Kukich K., Wolff S., Chi Lu, Chodorow M., Braden-Harder L. and Harris M.D.

Automated scoring using a hybrid feature identification technique. 1998.

Burstein J., Kukich K., Wolff S., Chi Lu, Chodorow M., Braden-Harder L. and Harris M.D. Computer analysis of essays. In *NCME Symposium on Automated Scoring*, 1998.

Burstein J., Leacock C. and Swartz R. Automated evaluation of essays and short answers. In *5th International Computer Assisted Assessment Conference*, Loughborough University. 2001

Callear D., Jerrams-Smith J. and Soh V. CAA of short non-MCQ answers. In *Proceedings of the 5th International CAA conference*, Loughborough. 2001. http://www.lboro.ac.uk/service/ltd/flicaa/conf2001/pdfs/k3.pdf.

Foltz P.W. Latent semantic analysis for text-based research. *Behavioral Research Methods, Instruments and Computers,* 28(2):197–202. 1996.

Foltz P.W., Britt M.A. and Perfetti C.A. Reasoning from multiple texts: An automatic analysis of readers' situation models. In *Proceedings of the 18th Annual Cognitive Science Conference*, pages 110–115. Hillsdale, NJ: Lawrence Erlbaum Associates. 1996.

Foltz P.W., Laham D. and Landauer T.K. Automated essay scoring: Applications to educational technology. 2003. http://www-psych.nmsu.edu/~pfoltz/reprints/Edmedia99.html. Reprint.

Graesser A.C., Wiemer-Hastings P., Wiemer-Hastings K., Harter D., Person N and the Tutoring Research Group. Using latent semantic analysis to evaluate the contributions of students in AutoTutor. *Interactive Learning Environments*, 8(2):87–109. 2000.

Jones M. and Jones G. *Advanced Biology*. Cambridge University Press. 2001.

Kanejiya D., Kumar A. and Prasad S. Automatic evaluation of students' answers using syntactically enhanced LSA. In *Building Educational Applications Using Natural Language Processing, Proc. of the HLT-NAACL 2003 Workshop*, pages 53–60. Association of Computational Linguistics. 2003.

Larkey L.S. Automatic essay grading using text categorisation techniques. In *ACM-SIGIR Inter. Conference on Research and Development inInformation Retrieval*. 1998.

Mitchell T. *Machine Learning*. McGraw Hill. 1997.

Mitchell T., Russell T, Broomhead P. and Aldridge N. Towards robust computerized marking of free-text responses. In *6th International Computer Aided Assessment Conference*. Loughborough. 2002.

Rosé C.P., Roque A., Bhembe D. and VanLehn K. A hybrid text classification approach for analysis of student essays. In *Building Educational Applications Using Natural LanguageProcessing*, pages 68–75. 2003.

Sleator D.K. and Temperley D. Parsing English with a link grammar. Technical Report. October 1991. CMU-CS-91-196.

Sleator D.K. and Temperley D. Parsing with a link grammar. In *Third International Workshop on Parsing Technologies*. 1993.

Wiemer-Hastings P. and Zipitria I. Rules for syntax, vectors for semantics. In *Proc. 23rd annual Conf. of the Cognitive Science Society*. Mawhwah, N.J. Erlbaum. 2001.

Wolfe M., Schreiner M.E., Rehder B., Laham D., Foltz P.W., Kintsch W. and Landauer T.K. Learning from text: Matching readers and texts by latent semantic analysis. *Discourse Processes*, 25(2 and 3):309–336. 1998.