Information Extraction and Machine Learning : Auto-Marking Short Free Text Responses to Science Questions[1]

Jana Z. SUKKARIEH and Stephen G. PULMAN
Computational Linguistics Group
University of Oxford, OX1 2HG, OXFORD, UK

Abstract Traditionally, automatic marking 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 computational linguistics 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. This paper describes the use of information extraction and machine learning techniques in the marking of short, free text responses of up to around five lines.

Keywords: Auto-marking, (self) assessment tool, information extraction, machine learning, computational linguistics, assessment in the service of learning.

[1] This is a 3-year project funded by the University of Cambridge Local Examination Syndicate.

Introduction

Recent advances in computational linguistics 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. E-rater developed by the Educational Testing service[2] ([2],[3],[4]) which uses shallow linguistic processing, and the Intelligent Essay Assessor (IEA) developed by Knowledge Analysis Technologies(KAT) [6] which uses latent semantic analysis are examples of (long) essay automatic marking systems. Our aim is to auto-mark free-text responses also, but only short ones of up to 5 lines, for content. E-rater and IEA do not work for such a task. E-rater depends, among other features, on the length of the essay, and IEA cannot tell the difference between "a student wrote an essay" or "an essay wrote a student". The responses we are dealing with are to factual science questions where there is an objective criteria for right and wrong, for example, the following GCSE biology answer:

Statement of the question	***Marking Scheme (full mark 2)***[3]
Baby polar bears use their mother's milk to keep them warm. ***Use your biological knowledge to explain how.***	***Any two from*** *Mother's milk is warm;Milk high energy content / lots of fat /lots of lactose/lots of sugar;Respiration to give energy/heat;Fat used for insulation;*

The system we have developed is experimental, designed to test the accuracy of the methods used. In a real setting, it is unlikely to be used as the sole marker in a high-stakes examination (partly because of legal sensitivies), but rather as an extra (completely consistent, stress and fatigue-proof) marker to check on the performance of human examiners. It could also be used in `formative' assessment, either for marking tests as a standalone system or as a part of a bigger one with a variety of short free text, multiple choice and graphically based questions. Such a system could be used as part of the learning process; students could use it for independent revision classes, or self-assessment or teachers could use it to free up time spent on marking.

From an initial random sample of data, we could tell that deep linguistic processing techniques were unlikely to work. We were also aware of the limitations of the shallow ones. However, we expected that information extraction and machine learning techniques were likely candidates for our application, since they do not require complete and accurate parsing and are relatively robust in the face of ungrammatical and incomplete sentences. Systems which tackle the same problem are being developed independently of ours. The most prominent among them are those developed by Leacock et al. [7] at ETS, Mitchell et al. ([8],[9]) at Intelligent Assessment Technologies and Rosé et al. [12] at Carnegie Mellon University. The 4 systems, ours included, are being developed independently, yet it seems they share similar characteristics. We will not go into any comparison here.

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

[3] X;Y/D/K;V is equivalent to saying that each of X, [L]={Y, D,K}, and V deserves 1 mark. The student has to write only 2 of these to get the full mark. [L] denotes an equivalence class i.e. Y, D, K are equivalent. If the student writes Y and D s/he will get only 1 mark.

In the following section, we briefly remind the reader on what information extraction is, we recapitulate the approaches we used and results described in Sukkarieh et. al ([13],[14]), and we also report on improvements we made since. In section 2, we remind the reader of some machine-learning techniques. We report on our experiments using such techniques and their corresponding results vis-a-vis our auto-marking problem. We conclude by summarising, describing the work-in-progress, and the tasks ahead. Due to the lack of space, we are, unfortunately, going to omit the statements of Biology questions and some interesting example answers that show the limitations of auto-marking and of shallow processing.

1. Information Extraction in a Nutshell

Information extraction (IE) techniques pull out pertinent information from a partially syntactically analysed text by applying a set of domain-specific patterns typically built from training data. In general, the information for filling a template may be found within a single sentence, across sequences of sentences, or sometimes in different forms several times within the same short text. Our auto-marking task is treated as information extraction from unstructured, free text, and where the templates may be matched across sentences and sometimes twice within the same passage.

The patterns or templates (we use the terms interchangeably here, although in some applications it makes sense to distinguish them) i.e., the rules that select from each text the information relevant to the task, are built from training data in one of the following ways. In each case we need to devise a language or a grammar to represent these rules. Before describing the methods and the results, we need to state which shallow linguistic properties we are considering and how we 'extract' them.

We have relied on part-of-speech tagging and information on noun phrases and verb groups in the data. We used a Hidden Markov Model part-of-speech (HMM POS) tagger trained on the Penn Treebank corpus, and a Noun Phrase (NP) and Verb Group (VG) 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. Shallow analysis makes mistakes, but multiple sources help fill gaps, and in IE this is adequate for most of the time. The general-purpose lexicon contains words with corresponding tags from the British National Corpus and the Wall Street Journal corpus. The domain-specific lexicon is obviously an on-going process.

Manually-Engineered Patterns

A person writes special knowledge to extract information using grammars and rules. The 3 crucial steps to take in writing extraction rules by hand can be found, among other references on information extraction, in Appelt and Israel (1999). First, all the ways in which the target information is expressed in a given corpus are determined. Second, all the plausible variants of these ways are considered and then written in appropriate patterns. We first describe the grammatical formalism with which we wrote the patterns. A pattern takes the form: *Id :: LHS ==> RHS*, where Id can be a complex term to categorise patterns into groups and subgroups. *LHS* is a *Cat*, where Cat is a (linguistic) category like NP, VG, Det, etc, or one that is user-defined. *RHS* is a list of *Elements,* where possibly each element is followed by a condition and Elements are defined:

Element ==> *Variable | Word/Cat | c(Cat)*
| ?(Element) optional element
| (Element; Element) disjunction
W(Word)

The first step in the pattern matching algorithm is that all patterns are compiled. Afterwards, when an answer arrives for pattern-matching it is first tagged and all phrases (i.e. verb groups-VG and noun phrases-NP) are found. These are then compared with each element of each compiled pattern in turn, until either a complete match is found or all patterns have been tried and no match was found to exist.

The grammar went through stages of improvement ([13],[14]), starting from words, disjunction of words, sequence of words, etc up until the version described above. We also experimented with a different number of answers used for the training data for different questions and, on average, we have achieved 84.5% agreement with examiners scores. Note that the full mark of each question range between 1-4.

Table 1. Results using the manually-written approach

Question	FullMark	Percentage of Agreement
1	2	89.4
2	2	91.8
3	2	84
4	1	91.3
5	2	76.4
6	3	75
7	1	95.6
8	4	75.3
9	2	86.6
Average	----	84

Table 1 shows the results using the last version of the grammar on 9 questions in the GCSE biology exams. For each question, we trained on 80% of the positive instances i.e. answers where the mark was > 0 (as should be done), and tested on the positive and negative instances. In total, we had around 200 instances for each question. The following results are the ones we got before we incorporated the spelling corrector into the system and before putting in place rules to avoid some over-generation. Also, we are in the process of fixing a few NP, VG formations and negations of verbs, and all this should make the percentages higher. However, as we will mention below, due to some inconsistency in the marking and the decisions that we had to make on what we should consider correct or not, independently of a domain expert, 84% average is a good result. Hence, though some of the results look disappointing, the discrepancy between the system and the examiners is not very significant. Furthermore, this agreement is calculated on the whole mark and not on individual sub_marks. This, obviously, makes the agreement worse than what in reality the pattern-matcher performance is. Finally, if more time were to be spent on tuning the patterns then also this result would be higher[4].

1.1 Automatic Pattern Learning

The last approach requires skill, much labour, and familiarity with both domain and tools. To save time and labour, various researchers have investigated machine-learning approaches to learn IE patterns. This requires many examples with data to be

[4] We have a demo available for the system.

extracted, and then the use of a suitable learning algorithm to generate candidate IE patterns. One family of methods for learning patterns requires a corpus to be annotated, at least to the extent of indicating which sentences in a text contain the relevant information for particular templates (e.g. [11]). Once annotated, groups of similar sentences can be grouped together, and patterns abstracted from them. This can be done by taking a partial syntactic analysis, and then combining phrases that partially overlap in content, and deriving a more general pattern from them. All that is needed is people familiar with the domain to annotate the text. However, it is still a laborious task. Another family of methods, more often employed for the named entity recognition stage, tries to exploit redundancy in un-annotated data (e.g. [5]). Previously, in [14], we said that we did not want to manually categorise answers into positive or negative instances, since this is a laborious task, and that we will only consider the sample of human marked answers that have effectively been classified into different groups by the mark awarded. However, in practise the noise in these answers was not trivial and, judging from our experience with the manually-written method, this noise can be minimized by annotating the data. After all, if the training data consists of a few hundred answers then it is not such a laborious task, especially if done by a domain expert. As we said previously, the idea is to automate (not fully) what we have been doing manually. The baseline algorithm is still the same.

Baseline

1. From the answer keys only, construct a set of sentences corresponding to correct answers.
2. *Automatically produce common syntactic and lexical variants of these. For example,*
   The wasps parasitise the caterpillars => The caterpillars are parasitised by the wasps.
   => The insects parasitise the larvae etc.
3. *Apply the resulting patterns to the training corpus.*
4. *Tweak if necessary and go back to step 4, or 5.Apply the resulting patterns to the testing corpus.*

A Supervised Learning or Semi-Automatic Algorithm The new algorithm omits the first 3 steps from the previously described learn-test-modify algorithm in [14]. In these 3 steps we were trying to automate the annotation task. We have replaced these steps with, hopefully a more reliable annotation done by a domain expert[5] and we start with the learning process directly. We keep the rest of the steps in the algorithm as they are, namely,

1. *The learning step (generalisation or abstracting over windows):*
   The patterns produced so far are the most-specific ones, i.e. windows of keywords only. We need some generalisation rules that will help us make a transition from a specific to a more general pattern. Starting from what we call a triggering window, the aim is to learn a general pattern that covers or abstracts over several windows. These windows will be marked as 'seen windows'. Once no more generalisation to the pattern at hand can be made to cover any new windows, a new triggering window is considered. The first unseen window will be used as a new triggering window and the process is repeated until all windows are covered (the reader can ask the authors for more details. These are left for a paper of a more technical nature).
2. *Translate the patterns (or rudimentary patterns) learned in step 1 into the syntax required for the marking process (if different syntax is used).*
3. *Expert filtering again for possible patterns.*
4. *Testing on training data. Make additional heuristics on width. Also, add or get rid of some initial keywords.*
5. *Testing on testing data.*

We continue to believe that the best place to look for alternatives, synonyms or similarities is in the students' answers (i.e. the training data). We continue in the process

[5] This does not mean we will not investigate building a tool for annotation since as it will be shown in section 2, annotating the answers has a significant impact on the results.

of implementation and testing . A domain expert (someone other than us) is annotating some new training data. We are expecting to report on these results very soon.

2. Machine-Learning Approach

In the previous section, we described how machine-learning techniques can be used in information extraction to learn the patterns. Here, we use machine-learning algorithms to learn the mark. Given a set of training data consisting of positive and negative instances, that is, answers where the marks are 1 or 0, respectively, the algorithm abstracts a model that represents the training data, that is, describing when or when not to give a mark. When faced with a new answer the model is used to give a mark.

Previously in [13], we reported the results we obtained using Nearest Neighbour Classification techniques. In the following, we report our results using two algorithms, namely, decision tree learning and Bayesian learning on the questions shown in the previous section. The first experiments show the results with non-annotated data; we then repeat the experiments with annotated data. The annotation is very simple: we highlight the part of the answer that deserves 1 mark, meaning that irrelevant material can be ignored. Unfortunately, this does not mean that the training data is noiseless since sometimes annotating the data is less than straightforward and it can get tricky. However, we try to minimize inconsistency. We used the existing Weka system [15] to conduct our experiments. In brief, the task of learning can be described as follows:

Given:

1. a set of attributes A1, … , An with corresponding values V(Ai) $\in \{Ø, V_{a1}, \ldots, V_{ai,}\}$. It is Ø when a value is missing. Values could be numeric, nominal, discrete or continuous
2. Possibly, categories or classes C1, …,Ct, Ø corresponding to the values of the outcome attribute
3. a set of instances, $\hat{\boldsymbol{I}}$, where each is a vector of the form Xj=(V(A1), … ,V(An)) and where Xj belongs or is assigned to a class Ck for some k
4. a new instance X=(V(A1),….V(An))

Aim:

The aim is to assign X to a class using a function (called also a concept or a model) that was **learned** from observing $\hat{\boldsymbol{I}}$. The task is then reduced to learning the concept. When categories are not given, clustering techniques are used to group instances that seem to fall together.

Each of the following learning algorithms learns the concept in a different way and has a different knowledge representation to represent it. We, briefly, describe each algorithm, describe how it can tackle our problem, and report the results. The results reported are on a 10-fold cross validation testing.

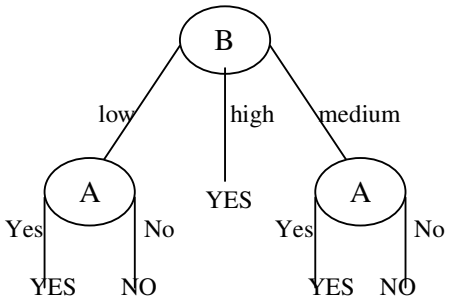
Decision Tree Learning(DTL)

For illustration, consider a very simple example. Assume the attributes are : A=**sunny** with values {yes, no}, C=**wind** with values {strong,weak} and B=**stress-at-work** with values {high, low, medium}. The outcome attribute is D=**go-to-the-gym** with values{YES, NO}.

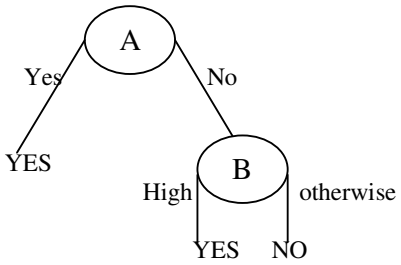
Instances :

Sunny	stress-at-work	Wind	Go-to-gym
Yes	High	Strong	YES
Yes	Medium	Weak	YES
No	Low	Strong	NO
No	High	Weak	YES
Yes		Weak	YES
No	Medium		NO

Example Tree:



However, there is another more compact tree with no replicated sub-trees that can describe the data and predict new assignments. Start with **sunny** as the parent node:



Learning Algorithm: Recursively select the "best attribute" to use at the current node in the tree. Once the attribute is chosen generate children nodes, one for each possible value of the selected attribute. Sort the instances according to the possible values of this attribute (each subset of the instances corresponds to a child node). Repeat this step for each node until all instances associated with a node are either all positive or negative. An instance is positive if it fits the concept description and negative if it does not.

Naïve Bayesian Learning

Bayesian learning assumes that all attributes are equally important. In reality, this assumption is not true but in practice it works well. Some of the reasons the classifier is so common is that it is fast, easy to implement, and relatively effective.

In our previous toy example, it answers a question like "what is the probability that one goes to the gym given some evidence, E about the stress at work, whether it is sunny or not, etc", i.e. find Pr(go-to-gym=yes| E). According to Bayes's rule of conditional probability (hence Bayesian Learning) and, naively, assuming that the pieces of evidence i.e. the rest of the attributes are independent (hence Naïve Bayesian; there are, techniques to eliminate, redundant non-independent attributes also),

$$Pr(yes|E) = \frac{\Pr(E1 \mid yes) \ldots \Pr(En \mid yes) \Pr(yes)}{\Pr(E)}$$

The problem then is reduced to estimating such probabilities from the instances given.

For our marking problem, the outcome attribute is well-defined. It is the mark for each question and its values are {0,1, …full_mark}. The input attributes could vary from considering each word to be an attribute or considering deeper linguistic features like a head of noun phrase or head of verb group to be an attribute, etc. In the following experiments, each word in the answer was considered to be an attribute.

Furthermore, (Rennie et al. 2003) propose simple heuristic solutions to some problems with naïve classifiers. In Weka, Complement of Naïve Bayes is supposed to be a refinement to the selection process that Naïve Bayes makes when faced with instances where one outcome value has more training data than another. This is true in our case. Hence, we ran our experiments using this algorithm also to see if there was any difference.

Results on Non-Annotated data

We first considered the non-annotated data, that is, the answers given by students, as they are. The first experiment considered the values of the marks to be {0,1, …, full_mark} for each question. The reports of decision tree learning and Bayesian learning are reported in the columns titled DTL1 and NBayes/CNBayes1. The second experiment considered the values of the marks to be either 0 or >0, i.e. we considered two values only. The results are reported in columns DTL2 and NBayes2/CNBayes2. The baseline is the number of answers with the most common mark over the total number of answers multiplied by 100. Obviously, the result of the baseline differs in each experiment only when the sum of the answers with marks greater than 0 exceeds that of those with mark 0. This affected questions 8 and 9 in Table 2 below. Hence, we took the average of both results. It was no surprise that the results of the second experiment were better than the first on questions with the full mark >1. After all, in the second experiment, the algorithm is learning a 0-mark and a symbol for just any mark>0 as opposed to an exact mark in the first. In both experiments, the Naïve Bayes learning algorithm did better than the decision tree learning algorithm and the complement of Naïve Bayes did slightly better or equally well on questions with a full mark of 1, like questions 4 and 7 in the table, while it resulted in a worse performance on questions with full marks >1.

Table 2. Results for Bayesian learning and decision tree learning on non-annotated data

Question	Baseline	*DTL1*	*NBayes/CNBayes1*	DTL2	NBayes/CNBayes2	Stem_DTL2	Stem_Nbayes2
1	69	*73.52*	*73.52 / 66.47*	76.47	81.17 / 73.52	--	--
2	54	*62.01*	*65.92 / 61.45*	62.56	73.18/ 68.15	--	--
3	46	*68.68*	*72.52 / 61.53*	93.4	93.95 / 92.85	--	--
4	58	*69.71*	*75.42 / 76*	69.71	75.42 / 76	--	--
5	54	*60.81*	*66.66 / 53.21*	67.25	73.09 / 73.09	--	--
6	51	*47.95*	*59.18 / 52.04*	67.34	81.63 / 77.55	73.98	80.10
7	73	*88.05*	*88.05 / 88.05*	88.05	88.05 / 88.05	93.03	87.56
8	42 / 57	*41.75*	*43.29 / 37.62*	72.68	70.10/ 69.07	81.44	71.65
9	60 / 70	*61.82*	*67.20 / 62.36*	76.34	79.03 / 76.88	71.51	77.42
Average	60.05	*63.81*	*67.97/62.1*	74.86	79.51/77.3	--	--

Since we were using the words as attributes, we expected that in some cases stemming the words in the answers would improve the results. Hence, we experimented with the answers of 6, 7, 8 and 9 from the list above and the results, after stemming, are reported in the last two columns in Table 2[6]. We notice that whenever there is an improvement, as in question 8, the difference is very little. Stemming does not necessarily make a difference if the attributes/words that make a difference appear in a root form already. The lack of any difference or worse performance may also be due to the error rate in the stemmer.

[6] Our thanks to Leonie Ijzereef for the results in the last 2 columns of Table 2.

Results on Annotated data

We repeated the second experiments with the annotated answers. As we said earlier, annotation means highlighting the part of the answer that deserves 1 mark (if the answer has >=1 mark), so for e.g. if an answer was given a 2 mark then at least two pieces of information should be highlighted and answers with 0 mark stay the same. Obviously, the first experiments could not be conducted since with the annotated answers the mark is either 0 or 1. The baseline for the new data differs and the results are shown in Table 3 below. Again, Naïve Bayes is doing better than the decision tree algorithm. It is worth noting that, in the annotated data, the number of answers whose marks are 0 is less than in the answers whose mark is 1, except for questions 1 and 2.

From getting the worse performance in NBayes2 before Annotation, Question 8 jumps to seventh place. The rest maintained the same position more or less, with question 3 always coming top. Count(Q,1)-Count(Q,0) is highest for questions 8 and 3, where Count(Q,N) is the number of answers whose mark is N. The improvement of performance for question 8 in relation to Count(8,1) was not surprising, since question 8 has a full-mark of 4 and the annotation's role was an attempt at a one-to-one correspondence between an answer and 1 mark. On the other hand, question 1 that was in seventh place in DTL2 before annotation, jumps down to the worst place after annotation. In both cases, namely, NBayes2 and DTL2 after annotation, it seems reasonable to hypothesize that P(Q1) is better than P(Q2) if Count(Q1,1)-Count(Q1,0) >> Count(Q2,1)-Count(Q2,0), where P(Q) is the percentage of agreement for question Q. Furthermore, according to the results of CNBayes in Table 2, we expected that CNBayes will do better on questions 4 and 7. However, it did better on questions 3, 4, 6 and 9. Unfortunately, we cannot see a pattern or a reason.

Table 3. Results for Bayesian learning and decision tree learning on annotated data

Question	Baseline	DTL	NBayes/CNBayes
1	58	74.87	86.69 / 81.28
2	56	75.89	77.43 / 73.33
3	86	90.68	95.69 / 96.77
4	62	79.08	79.59 / 82.65
5	59	81.54	86.26 / 81.97
6	69	85.88	92.19 / 93.99
7	79	88.51	91.06 / 89.78
8	78	94.47	96.31 / 93.94
9	79	85.6	87.12 / 87.87
Average	69.56	84.05	88.03 / 86.85

As they stand, the results of agreement with given marks are encouraging. However, the models that the algorithms are learning are very naïve in the sense that they depend on words only and providing a justification for a student won't be possible. The next step is to try the algorithms on annotated stemmed[7]data that has been corrected for spelling and investigate some deeper features or attributes other than words, like the heads of a noun phrase or a verb group or a modifier of the head, etc. 88% seems high for such a naïve method but recall that this was a supervised learning in the sense that the data was manually annotated.

3. Conclusion

In this paper, we have described the latest refinements and results made on our automarking system described in ([13],[14]), using information extraction techniques where patterns were hand-crafted or semi-automatically learned. We have also described experiments where the problem is reduced to learning a model that describes

[7] We now have a more accurate stemmer.

the training data and use it to mark a new question. At the moment, we are focusing on information-extraction techniques. The results we obtained are encouraging enough to pursue these techniques with deeper linguistic features, especially to be able to associate some confidence measure and some feedback to the student with each answer marked by the system. We are using machine-learning techniques to learn the patterns or at least some rudimentary ones that the knowledge engineer can complete. As we mentioned earlier in section 1.2, this is what we are in the process of doing. Once this is achieved, the next step is to try and build a tool for annotation and also to use some deeper linguistic features or properties or even (partially) parse the students' answers. We have noticed that these answers vary dramatically in their written quality from one group of students to another. For the advanced group, many answers are more grammatical, more complete and with less spelling errors. Hence, we may be able to extract linguistic features deeper than a verb group and a noun group.

Bibliography

[1] Appelt, D. and Israel, D. (1999) Introduction to Information Extraction Technology. IJCAI 99 Tutorial.

[2] 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.

[3] 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.

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

[5] Collins, M. and Singer, Y. (1999) Unsupervised models for named entity classification. Proceedings Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, pp. 189-196.

[6] 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.

[7] Leacock, C. and Chodorow, M. (2003) C-rater: Automated Scoring of Short-Answer Questions. Computers and Humanities 37:4.

[8] Mitchell, T. Russel, T. Broomhead, P. and Aldridge, N. (2002) Towards robust computerized marking of free-text responses. In 6th International Computer Aided Assessment Conference, Loughborough.

[9] Mitchell, T. Russel, T. Broomhead, P. and Aldridge, N. (2003) Computerized marking of short-answer free-text responses. Paper presented at the 29th annual conference of the International Association for Educational Assessment (IAEA), Manchester, UK.

[10] Rennie, J.D.M., Shih, L., Teevan, J. and Karger, D. (2003) Tackling the Poor Assumptions of Naïve Bayes Text Classifiers. http://haystack.lcs.mit.edu/papers/rennie.icml03.pdf.

[11] Riloff, E. (1993) Automatically constructing a dictionary for information extraction tasks. Proceedings 11th National Conference on Artificial Intelligence, pp. 811-816.

[12] Rose, C. P. Roque, A., Bhembe, D. and VanLehn, K. (2003) A hybrid text classification approach for analysis of student essays. In Building Educational Applications Using Natural Language Processing, pp. 68-75.

[13] Sukkarieh, J. Z., Pulman, S. G. and Raikes (2003) N. *Auto-marking: using computational linguistics to score short, free text responses*. Paper presented at the 29th annual conference of the International Association for Educational Assessment (IAEA), Manchester, UK.

[14] Sukkarieh, J. Z., Pulman, S. G. and Raikes (2004) N. *Auto-marking2: An update on the UCLES-OXFORD University research into using computational linguistics to score short, free text responses*. Paper presented at the 30th annual conference of the International Association for Educational Assessment (IAEA), Philadelphia, USA.

[15] Witten, I. H. Eibe, F. *Data Mining.* Academic Press 2000.