Building a Textual Entailment Suite for the Evaluation of Automatic Content Scoring Technologies

Jana Z. Sukkarieh, Eleanor Bolge
Educational Testing Service
Rosedale Road, Princeton, NJ 08541
E-mail: jsukkarieh@ets.org, ebolge@ets.org

Abstract

Automatic content scoring for free-text responses has started to emerge as an application of Natural Language Processing in its own right, much like question answering or machine translation. The task, in general, is reduced to comparing a student's answer to a model answer. Although a considerable amount of work has been done, common benchmarks and evaluation measures for this application do not currently exist. It is yet impossible to perform a comparative evaluation or progress tracking of this application across systems – an application that we view as a textual entailment task. This paper concentrates on introducing an Educational Testing Service-built test suite that makes a step towards establishing such a benchmark. The suite can be used as regression and performance evaluations both intra-c-rater® or inter automatic content scoring technologies. It is important to note that existing textual entailment test suites like PASCAL RTE or FraCas, though beneficial, are not suitable for our purposes since we deal with atypical naturally-occurring student responses that need to be categorized in order to serve as regression test cases.

1. Introduction

Automatic content scoring for free-text responses has started to emerge as an application of Natural Language Processing (NLP) in its own right, much like question answering or machine translation. There has been a considerable amount of work done e.g., (Christie, 1999, 2003; Larkey et al., 1998; Rehder et al., 1998; Wiemer-Hastings et al., 1999; Ming et al., 2000; Callear et al., 2001; Rudner & Liang, 2002; Mitchell et al., 2002; Laham et al., 2002; Mason & Grove-Stephenson, 2002; Leacock & Chodorow, 2003; Foltz et al., 2003; Rosé et al., 2003; Sukkarieh et al., 2003; Datar et al., 2004 ;Williams & Dreher, 2004; Siddiqi & Harrison, 2008; Moehler & Mihalcea, 2009 and Sukkarieh & Blackmore, 2009). However, common benchmarks and evaluation measures for this application do not currently exist. It is yet impossible to perform a comparative evaluation or progress tracking of this application across systems.

Existing systems, including c-rater®, the Educational Testing Service (ETS) technology for the automatic content scoring of short free-text responses, report results in terms of the scoring agreement between human raters and systems. There is no common measure used to make scoring results comparable. Scoring agreement has been reported in terms of exact or adjacent percentages, Pearson or Spearman's correlation, standard deviation, F-score (precision, recall), false positives/negatives, and kappa statistics. In addition, no large-scale common benchmark evaluation set of items exists with which to compare the results, due mainly to intellectual property issues. Furthermore, score results are neither very meaningful (from an NLP point-of-view) nor give any sense of developmental progress. They may give an indication that one version of the system is agreeing with human raters better than another but verifying that this occurs for the right reasons often means hand-checking hundreds of cases. Consequently, we have built an evaluation suite which serves as a diagnostic comparative and performance evaluation mechanism. In this paper, we concentrate on introducing this ETS-built evaluation suite that makes a step towards establishing such a benchmark which can be used as regression test suite both intra-c-rater® or inter automatic content scoring technologies. Note that this is ongoing work that will benefit from discussions and comments within the automatic content-scoring community.

We will first examine the task of content scoring for free-text responses and its relationship to textual entailment with a high-level description of c-rater®. Next, we will concentrate on presenting the test suite by describing the requirements, principles or guidelines used, and what was built at various stages. We also emphasize the difference between the ETS-suite and existing textual entailment suites. Finally, we will outline a future plan on how our effort could mature to advance the automatic content scoring for free-text responses.

2. Automatic Content Scoring

We consider "content"[1] to be a set of main points or concepts predefined by a test developer. Such set forms the evidence for knowledge which a student needs to demonstrate in a response. Table 1 shows an example of an item[2] with the required content for the response (as shown by the concepts or main points $\mathbf{C_1}$, $\mathbf{C_2}$ and $\mathbf{C_3}$) and the recommended rubric for assigning score points. These score points will be referred to as total scores in the remainder of this paper. This view of "content" is not necessarily restricted to short-response items. The task of knowing whether a set of main points or concepts are conveyed by a certain text and providing feedback for a student will have to deal with the same issues reported here, regardless of the length of the text or the techniques used to solve this task.

[1] A survey of the literature of automatic content scoring reveals that "content" means different things to different researchers/organizations. Elaborating on this is better left for another occasion.

[2] An item at ETS refers to a question appearing in either a test or an exam.

Table 1. A test item with the required concepts

Item (Full credit: 2 points) _Stimulus_: A reading passage _Prompt:_ In the space provided, write the question that Alice was most likely trying to answer when she performed Step *B*.	_Concepts or main points:_ **C_1:** How does rain formation occur in winter? **C_2:** How is rain formed? **C_3:** How does temperature contribute to the formation of rain?
Scoring rules: 2 points for **C_1** 1 point for **C_2** (only if **C_1** is not present) or for **C_3** (only if **C_1** and **C_2** are not present) Otherwise 0 points	

We view the task of automatic content scoring as a textual entailment task as follows:

Given a concept, *C*, (e.g., C3 in Table 1 "How does temperature contribute to the formation of rain?") **and** a student response, *A*, (e.g., either "How does temperature assist in the formation of rain?" or "Does temperature affect the way altitude helps in rain formation?") **and** the context of the item, **the goal is** to check whether *C* is an ***inference*** or ***paraphrase*** of *A* (in other words, *A* implies *C* and *A* is true).

The c-rater® technology at ETS tries to tackle such a task.

2.1 c-rater® in brief

There are four main steps in c-rater® (See Figure 1). The first is what we call item-dependent Model Building (MB), where a set of model responses are generated for the item at-hand guided by a set of scored student responses and a set of lexical resources generating **similar lexicon**. A scored student response is one where a human rater highlights what s/he considers to be the portion of the response that entails a concept labeling the <Response, Concept> pair with "Present" or the portion that refutes a concept while labeling the pair by "Refuted", otherwise s/he says the response does not entail the concept and labels the pair "Absent". We call {Absent, Present, Refuted} analytic-based scores. The highlighted portion corresponding to an entailment is called positive evidence and the one corresponding to a refutation is called negative evidence. Second, c-rater® automatically processes the model responses and students' responses using a set of NLP tools and resources. In the process, it extracts a set of linguistic features. This action consists of the following: a response is processed for **spelling corrections** in an attempt to decrease the noise for subsequent NLP tools. In the next stage, **tokenization**, **parts-of-speech tagging** and **parsing** are performed (the OpenNLP parser is used http://opennlp.sourceforge.net). After that, a parse tree is passed through a **feature extractor** where manually-generated rules extract features from the parse tree and introduce semantic roles. Next, a **pronoun resolution** stage is performed where pronouns are resolved to either an entity in the student's response or the question. Finally, a **morphology analyzer** reduces the words to their lemmas.

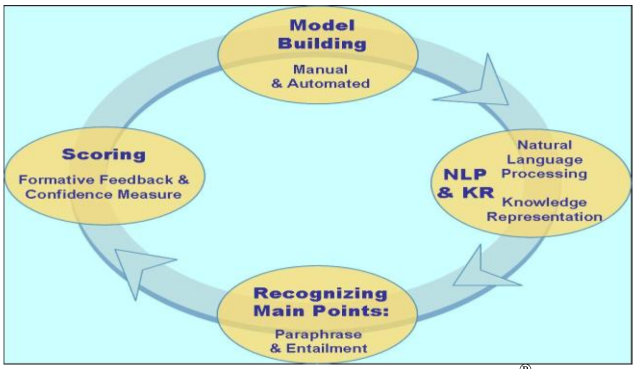


Figure 1. The four main steps in c-rater®

Third, a **concept-detector** uses the linguistic features culminated from both MB and NLP to automatically determine whether a student's response entails the predefined concepts.

Finally, c-rater® applies the scoring rules to produce a total score and feedback that justifies the total score to the student.

3. A Textual Entailment Test Suite

3.1 Regression Testing in NLP-based applications

The metrics for measuring the functionality of a certain technology include measuring progress, comparing one version to another, and monitoring the effect of frequent modifications. We believe that this proves particularly challenging due to the nature of the NLP-based application and the automatic content scoring task. In most software, the business value is in a working product. In automatic content-scoring technologies it is not only about producing a total score but producing one for the "right" reasons – not due to errors in the linguistic features obtained. For example, in high-stakes testing, a lawsuit could be incurred from wrongly-justified scores. Until recently, comparing versions of c-rater® meant comparing its total scores to the original human scores without a sense of the effect of particular changes. Evaluating the effect of a change often meant hand-checking hundreds of cases.
To improve monitoring, we have designed a regression test suite and introduced automated testing. This produces a finer-grained view of the modules' performances, increases our confidence about the "correctness" of our scores and most importantly provides guidance for product development.

There is considerable previous work on grammar development strategies and test suites for grammar-based systems. For example, Prasad & Sarkar (2000) have compared corpus-based and test-suite based coverage for grammar while de Paiva & Holloway King (2008) have built an entailment suite and used it for a question answering application. However, we do not find regression test suites that consist of **categorized**

naturally-occurring atypical data[3].

Atypical data includes noise, unconventional textual representation and mixed-mode representation. Noise includes, among others, incomplete sentences, misspellings, ungrammaticality and random keyboard indefinite stroking of the same letter. Noise varies from one grade level to another, from one population to another and from one content area to another. Unconventional textual representation includes, among others, symbols, SMS abbreviations, foreign and slang words. Furthermore, some content areas require students' responses in mixed-mode: visual, textual and mathematical symbolic language. To date, we only considered a mixed-mode representation of text, mathematical symbols and equations.

One would definitely start with having "typical" or representative well-written English test cases but one cannot expect anything "typical" in students' responses. Hence, a test-suite of naturally-occurring atypical test cases is needed for a system like an automatic content scoring technology because the robustness of (or lack thereof) NLP tools towards atypical data needs to be taken into account. Though one cannot account for all possibilities of potentially atypical data, one may be able to predict or learn a certain model of "behavior" with particular types. Consequently, one may be able to preempt a behavior by either pre-processing the data, backing some tool up to overcome the behavior, or even ignoring it when it has no effect on the final output.

The existing textual entailment suites were not suitable for our purposes. In particular, the commonly-used Pascal RTE suites (http://pascallin.ecs.soton.ac.uk/Challenges/RTE) were not suitable enough for our purposes because of the following:

1. As mentioned above we deal with student responses; hence, data is "atypical". The RTE data (through RTE-4) consists of correctly-written English.
2. Our application is different from the focus of RTE to date. "…even though different applications need similar models for semantic variability, the problem is often addressed in an application-oriented manner and methods are evaluated by their impact on the final application" (Dagan et al., 2005).
3. Most importantly, we require a systematic diagnostic evaluation that guides the development of our product something that the RTE data does not provide.

Furthermore, the FraCas textual entailment test suite (Cooper et al. 1996) is not suitable enough either. Even though the suite is categorized into various phenomena and it has been used to guide the development of a Knowledge Representation and a deductive textual entailment engine (Sukkarieh, 2001), the suite is built from artificial test cases of well-written English. However, we plan to use the FraCas categorization for further extension of c-rater's semantic capability.

3.2 c-rater® Test Suite: Description

The suite is built from naturally-occurring real student responses collected from various assessment programs and varying content areas that c-rater® has processed. It currently has about 350 items.

The c-rater® suite consists of 6-tuples in the form of *<Test_Id, Text, Hypothesis, Human_Label, c-rater_Label, Category>* that can be further extended by the user to a 7-tuple suite where the 7th argument is *List_of_Modules_Outputs* which will be defined in section 3.3.3. In the remainder of the paper, we will refer to a tuple in the test suite by an "engine test".

Test_Id is a unique id given to the engine test. *Text* and *Hypothesis* are extracted from c-rater's database. They will be further specified in section 3.3. A "*Human_Label*" is an analytic-based score i.e. either *Present*, *Refuted*, or *Absent*. Initially a *c-rater_Label* for each engine test is *"Absent"* which gets replaced by the analytic score that c-rater® assigns automatically to the test. A *Category* can be any of the following kind.

3.2.1 Linguistic Phenomenon

The suite is mostly driven by linguistic phenomena that exist in the data, i.e., each linguistic phenomenon constitutes the criterion that "decides" whether a text entails a certain concept, does not entail or refutes it. For example, an ergative verb is the criterion for which *"you could heat bricks"* could entail the concept "*your bricks could heat"*. The phenomenon could be general too e.g. "implicit negation" is the criterion for which *"clouds prevented him from seeing the moon"* refutes *"he can see the moon"*. More than one phenomenon could be at play when deciding about an entailment.

3.2.2 A module under c-rater's hood

Another issue that could motivate the inclusion of an engine test in the suite is an unexpected output of an NLP module under the hood of c-rater® (including the spelling corrector and the concept-detector). This either means that the response was well-written but a module's output was unexpected or that the response was noisy and hence the module did not recover and produced wrong output that affected the decision of the concept-detector.

3.2.3 Mixed-Mode Representation

This refers to a mixture of modes of representation of a student response. As we mentioned earlier, to date we only deal with responses consisting of mixed textual and mathematical equations or symbols.

3.2.4 Unconventional Textual Representation

This, as mentioned earlier, refers to all kinds of "correct natural language" but unconventional e.g. foreign lexicon embedded in English and SMS abbreviations.

[3] For lack of finding a better nomenclature we use "atypical".

An engine test could belong to more than one *Category*. In the following section, we provide more details for the arguments of the engine tests as we progressed with them. Note that this is not a description on how c-rater® was developed or what can or cannot do. The following is the progression of thoughts about building a regression test suite.

3.3 c-rater® Test Suite: Stages, Methodology and Types of Engine Tests Built at Each Stage

3.3.1 Stage 1

The principle is simple: "we require a set of engine tests that will make sure that the decision of the concept-detector does not change from one version to another." The method we followed also was simple. Select a few hundred pairs of <*Positive_Evidence*, *Concept*> from scored student responses across all items. Hence, the *Positive_Evidence* is either a complete response or part of a student response that entails the *Concept*. The engine tests in Stage 1 looked as follows:

<*Test_Id, Positive_Evidence, Concept, Human_Label, c-rater_ Label*>

where *Human_Label* is "*Present*" for all of our test cases. The testing condition was again simple "If *Human_Label* ≠ *c-rater_Label* then *Test_Id* is flagged automatically" for a human's attention. At this stage also, representation in the suite of morphological variations that do not modify the part-of-speech (POS) tags was made.

3.3.2 Stage 2

When *Test_Id* is flagged in Stage 1, the question was then: Did c-rater® fail because we failed to take into account some linguistic phenomenon and if so, which one? Also, if we are concerned about doing the right thing then we ought to ask whether c-rater® agrees with the human raters for the right reasons or is it just a guess, a mistake, etc.

Basically, we ask questions about "coverage" just as is done in developing parsers and grammar-based test suites (Prasad and Sarkar, 2000): a) how many linguistic phenomena does the technology handle? And, b) How many responses in naturally-occurring student response corpora does the technology handle? The method we followed was as follows:

1. Select a syntactic phenomenon of interest (*Category*)
2. Select a concept as given by the rubric of a test item or a positive evidence for that concept (*Hypothesis*)
3. Select a naturally-occurring student response or part of a response (*Text*) such that *Text* entails *Hypothesis* due to *Category*
4. Add engine tests such that *Text* in <*Text*, *Hypothesis*, *Category*> considered in 1, 2 and 3 is injected manually with some variations where *Injected_Text* does not entail *Hypothesis* (*Injected_Text* is *Text* with the injected variations)

In Stage 2, once the "Phenomenon" of interest is selected then either:

a) It is already covered by the feature extractor and needs to be tested and in that case we use the feature extractor to guide us into finding the relevant student responses and a human rater just verifies them
b) It is not covered by the feature extractor and in that case the human rater uses the output of the parser for some guidance in order to find the relevant student responses in our database

Consequently, the engine tests looked as follows in Stage 2, where *Category* is a syntactic phenomenon (including derivational morphology with a change of POS):

<*Test_Id, Text, Hypothesis, Human_Label, c-rater_ Label, Category*>

The engine tests under each category are further categorized, similarly to the ones found in de Paiva & Holloway King (2008) i.e. in order of processing complexity. Some examples follow. The *Hypothesis* is underlined for clarity.

Type 1: Sanity Check Engine Tests

These are entailments that look too trivial not to perform. However, one should emphasize, these are not as trivial as they look when dealing with noisy data.

(1) *<Test_Id1, "The animal is infected", "The animal is infected", Present, _, Identical>*

Type 2: Single Phenomenon Single Sentence Engine Tests

These are engine tests where both the *Hypothesis* and the *Text* consist of single sentences and where the entailment is due to a single phenomenon.

(2) *<Test_Id2, "The bill should not be passed because psychologists do not have the training of medical doctors to know when drugs should and should not be prescribed, how different drugs work together, what types of side effects occur, and how to deal with these effects when they do occur.", "Psychologists are not trained", Present, _, Nom_to_Verb>*

where *Nom_to_Verb* denotes "nominalization to tensed clause" defined by Vanderwende et al. (2006).

Type 3: Single Phenomenon Multi-Sentence Engine Tests

These are engine tests where either the Hypothesis or the Text consists of multi-sentences and where the entailment is due to a single phenomenon.

(3) *<Test_Id3, "The fish populations will proball decreas a lot. If they constantly have to breath likd that then it will over stress their body killing them", "*<u>*This will decrease the fish populations*</u>*", Present, _, Ergative>*

<u>**Type 4: Multi-Phenomena Single Sentence Engine Tests**</u>
These are engine tests where an entailment is due to more than one phenomenon and both *Text* and *Hypothesis* are single sentences. Such an engine test will appear under more than one *Category*.

(4) *<Test_Id4,"The gasses make the fish fight for air and make the fish needs to breathe more fast to get more oxygen than before. ", "*<u>*The gas makes the fish need to breath faster to get more oxygen*</u>*", Present,_, Category>*

Ignoring that there is a need for spelling-correction, there is a need to at least a) recognize that gases and gas have the same lemma, b) distribute the conjunction so that "the gases make the fish …" and c) comparative use i.e. recognize that "more fast" and "faster" are the same.

<u>**Type 5: Multi-Phenomena Multi-Sentence Engine Tests**</u>
These are tests where an entailment is due to more than one phenomenon and either *Text* or *Hypothesis* consists of multi-sentences.

(5) *<Test_Id5, "It is supposed to show that presient Johnson knows how to do the job and that he wants to fix the problems for the common worker and American. It also shows how Gladwater believes that draft si a waste and that people who join voluntarily join the military will be better then those who are forced to", "*<u>*Gladwater believes people should join the army voluntarily*</u>*", Present,_, Category>*

At least, the distributive property and the properties of dependent/relative clauses are at play in *Test_Id5*.

<u>**Type 6: Manually-Injected Variations of Engine Tests**</u>
As mentioned earlier, we also, manually, inject *Text* in some engine tests with some variations for their entailment to fail. These were devised purposely to avoid false positives. An example under the passives *Category* follows.

(6) *<Test_Id6,"The animal was infected by the doctor", "*<u>*The animal infects the doctor*</u>*", Absent, _, Passives>*

where the original *Text* is: "The doctor was infected by the animal".

336 engine tests were devised at this stage. The *Hypotheses* in all the tests were single sentences. However, we have considered an initial set of around 800 responses that the 336 tests were selected from.

3.3.3 Stage 3

As textual entailment obviously is not due to syntactic phenomena only, at this stage one requirement was not only to extend the kinds of syntactic categories considered in Stage 2 but also to include "lexical semantics" categories and "semantics beyond lexicon" in the test suite. We also needed to start categorizing the negative evidence or why a *Hypothesis* is *Refuted*. Verifying whether one or more phenomenon is at play becomes a more daunting task. Consequently, another requirement was to have two human annotators instead of one as it was the case up until Stage 2. Furthermore, an additional requirement over Stage 2 included another kind of semantics for a *Category*; one which deals with an unexpected output by any pre-processing tool or any of the NLP tools (e.g. the parser). Finally, we needed to automate parts of the process or make it less demanding for humans.

A *Category* at this stage then can be a) one considered at Stage 2, b) a name of a tool/module *X* meaning "unexpected output of tool *X*". X can be, e.g., pre-parser, parser, pronoun-resolver, feature-extractor, or concept-detector or c) an extended set of linguistic phenomena that we describe next.

The phenomena for "*Present*" are divided into "Syntactic," "Lexical," and "Semantics beyond lexicon." The syntactic categories include phenomena like "Passives," "Ergative," "Partitives," "Possessives," "Comparatives and Superlatives," "Phrasal Verbs," "Appositives," "Dependent Clauses other than appositives," "Interrogatives", "Extraposition," and "Adverb final and non final". Some additional categories were driven by those in Vanderwende et al. (2006) like "Nominalization to Tensed Clause" and "Finite to Non-finite Constructions." We also have a category "None of the syntactic categories above."

Lexical categories include phenomena like "Exact Lexical Overlap," "Direct Synonymy Replacement" (not including compound synonymy), "Compound Synonymy," "Lexical Inference," and "Compounds_Other." So far, there is only one category labeled "Semantics_Beyond_Lexicon."

Engine tests with *Human_Label* of *"Refuted"* are categorized to date into three categories: "Explicit Negation," "Implicit Negation," and "Contradictory Information (other than negation)."

Engine tests with labels of "*Absent*" are not categorized.

The selection of a certain category is often guided by the rubrics of a certain item. For example, if in a particular pilot we noticed that all items require some kind of proportional reasoning which translates into using some sort of comparatives then concentration on "comparatives" takes priority.

The engine tests have the following format at this stage:

<*Test_Id, Text, Hypothesis, Human_Label, c-rater_ Label, Category, List_of_Modules_Outputs*>

where *List_of_Modules_Outputs* is optionally displayed and consists of the list of these self-explanatory elements:

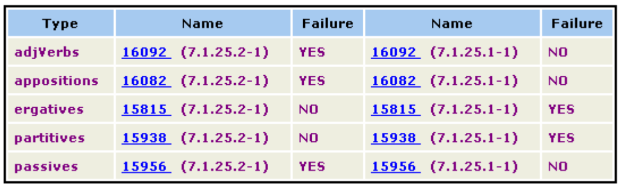
[Text_after_Spelling_Correction, Hypothesis_after_Spelling_Correction, Text_Parser_Output, Hypothesis_Parser_Output, Text_Feature_Extractor_Output, Hypothesis_Feature_Extractor_Output, Text_Morphology_Module_Output, Hypothesis_Morphology_Module_Output, Concept_Detection_Module_Output]

The method followed to generate the suite is:

1. Build an annotation tool that gets fed automatically from the c-rater® database to facilitate the work for a human rater.
2. Extract automatically a random set of 1600 pairs of student responses and concepts or model responses from the c-rater® database. No reason for selecting 1600 except that it is double the size considered when devising engine tests for fewer categories. Using only 1600, there is no guarantee there will be a balance in the representation of categories[4] or the *Human_Label*. However, we have a large size of additional data in our database in order to reach a balance.
3. Two humans are asked to annotate the engine tests. A human rater is to click one of three radio buttons to provide a *Human_Label* and click on one or more radio buttons where each button corresponds to a *Category*. If a *Text* is more than one sentence long then the annotator is provided first with <*Sentence*, *Hypothesis*> pairs for each sentence and then with <*Text*, *Hypothesis*> for annotation. An adjudicated annotation by a third human is used when there is a disagreement. When the three human raters cannot decide on a given engine test, it is discarded and replaced by another pair.

We have so far completed tasks 1 and 2 above. Task 3 is still in progress but a training step towards it was completed. Two human raters were asked to label a random set of 330 engine tests without consulting each other and without selecting a *Category*. They disagreed on 107 engine tests. A discussion followed this phase and the fact is: it is a non-trivial task particularly because the context of the item is not provided with the engine tests.

[4] Additional categories might be noted too.



Type	Name	Failure	Name	Failure
adjVerbs	16092 (7.1.25.2-1)	YES	16092 (7.1.25.1-1)	NO
appositions	16082 (7.1.25.2-1)	YES	16082 (7.1.25.1-1)	NO
ergatives	15815 (7.1.25.2-1)	NO	15815 (7.1.25.1-1)	YES
partitives	15938 (7.1.25.2-1)	NO	15938 (7.1.25.1-1)	YES
passives	15956 (7.1.25.2-1)	YES	15956 (7.1.25.1-1)	NO

Figure 2. A snapshot of comparative results

In addition, one human rater painstakingly looked at a random list of responses in c-rater's database where the total scores generated by c-rater® were different from the original human scores and annotated 120 engine tests for the *Category* meaning "*tool X*" mentioned earlier. In the following, we report some statistics corresponding to engine tests built at Stage 2 and in this last step of Stage 3.

4. Score Report

The test suite is used for regression testing i.e. the systematic diagnostic and comparative evaluation for c-rater's performance. The task here is to maintain the consistency of c-rater®. When new additions or modifications are made to c-rater®, we automatically verify that earlier analyses have not been contradicted. Currently, any change to the agreement/disagreement between *Human_Label* and *c-rater_Label* from one version to another is flagged automatically and corresponding engine tests are displayed for a human to verify. The suite helps us identify missing phenomena, which phenomena c-rater® fails to capture, and account for rare phenomena (similar to parsers' evaluations).

A web-based report is produced automatically when a new version of the system is built. Figure 2 shows a snapshot of the display when the results of the engine tests change.

Only the *Category* where a change occurs is shown with a list of *Test_Ids* whose results have changed. In the figure, only one test has changed under each *Category*. The change can be seen in the values of the **Failure** column i.e. {*YES*, *NO*}. YES means *c-rater_Label* ≠ *Human_Label* and NO means they are the same. A human can click to see the engine test in details. The version numbers are displayed too (in the figure 7.1.25.1-1 and 7.1.25.2-1 are compared).

Another use for the suite is benchmark performance evaluation for the same version of the system. The question here is for how many engine tests extracted from naturally-occurring corpora does c-rater® (and potentially, all other similar technologies) produce a correct decision? This is evaluated in terms of agreement with a human rater. Some statistics like: quadratic kappa statistics, confusion matrices, precision and recall are produced automatically to represent this agreement[5].

[5] Performance evaluation over the total scores of a set of approximately 350 benchmark items is also performed.

In total, we will report results on 456 engine tests. First, it is worth mentioning the following statistics for these tests:

	Hypothesis	Text
Avg. # Sentences per test	1.00	1.49
Avg. #Tokens per test	7.65	26.86
Avg. #Tokens per test w/out end punctuation	6.64	25.38

Table 2 shows the results of agreement between c-rater and the human rater in terms of confusion matrices[6] over some of the syntactic categories we have to date. The matrix represents:

		c-rater	
		Absent	**Present**
Human	**Absent**	N1	N2
	Present	N3	N4

AdjVerbs denotes the verbs considered as adjectives like "infected". We assume that the reader is familiar with the rest of the categories in the table.

Table 2. c-rater_Label vs. Human_Label for Phenomenon

Category	#Tests	#Failure	Confusion Matrix	
AdjVerbs	14	0	0	0
			0	14
Appositives	14	3	0	0
			3	11
Comparatives	38	28	6	28
			0	4
Dependent Clauses	73	28	3	6
			22	42
Ergative	122	58	51	28
			30	13
Nom_to_Verb	9	5	0	2
			3	4
Distributive Property	22	8	2	0
			8	12
Mixed-Mode	6	2	4	0
			2	0
Negation	6	2	4	0
			2	0
Partitives	11	3	2	1
			2	6
Passives	21	3	2	1
			2	16
Total	336	140	74	66
			74	122

There are 140 engine tests labeled "Absent" by a Human and 196 labeled "Present". c-rater fails to agree with the Human on 140 engine tests.

[6] Though there is enough evidence that kappa statistics is the best measure to evaluate agreement with humans, kappa statistics are not meaningful for such a small size of engine tests built so far. Hence, we only display confusion matrices.

Table 3 shows the results of agreement between c-rater and the human rater for some of the "tool X" categories.

Table 3. c-rater_Label vs. Human_Label for "tool X"

Category	#Tests	#Failure	Confusion Matrix	
Concept-detector	71	10	21	2
			8	40
Feature_Extractor	17	1	0	0
			1	16
Pronoun_Resolver	2	2	0	0
			2	0
Parser	27	12	1	1
			11	14
Pre-Parser	3	1	1	0
			1	1
Total	120	26	23	3
			23	71

c-rater fails to agree with the human rater on 26 engine tests. When a larger size suite is built, agreement between humans and c-rater will be reported in terms of quadratic kappa statistics.

5. Summary and Future Plans

ETS is a leading organization in educational assessment. Automatic content scoring is a fast-growing application of NLP. We have described a test suite that depends on the performance of c-rater. This work is in progress and more results will be shared in the future. There are many questions to deal with e.g. how to automate further parts of the process as to make it less demanding to generate engine tests and if that was the case how does one ensure balance between *"Absent", "Present" and "Negated"* under each category?

In addition, a lot can be drawn from work done by the grammar development researchers and the FraCas categories to further our development and regression evaluation. We would like to look for opportunities where we can share the use of the annotation tool and the test suite[7] as well as collaborate with other researchers so that we can collectively agree and progress can be tracked and compared. We would like to foster collaboration among researchers and organizations that are specifically working on automatic content scoring.

Furthermore, adequacy evaluation is important, i.e., clients should be able to compare available technologies and decide which technology is best for their purposes or add value to their practice. For example, teachers at schools may want to verify how adequate a technology is to score and provide feedback on students' homework. Hence, we need to look into ways to allow a client to perform such an evaluation.

[7] As long as neither the items nor the context of the items are shared without the permission of the assessment programs at ETS.

6. References

Callear, D., Jerrams-Smith, J. & Soh, V. (2001) CAA of short nonMCQ answers. In the Proceedings of the 5th International Computer Assisted Assessment Conference, Loughborough.

Cooper, R., Crouch, D., van Eijck, J., Fox, C. van Genabith, J., Jaspars, J., Kamp, H., Milward, D., Pinkal, M., Poesio, M. & Pulman, S. (1996). The FraCas Consortium, Deliverable D16. With additional contributions from Briscoe, T., Maier, H. and Konrad, K.

Christie, J. R. (1999). Automated essay marking for both content and style. In Proceedings of the 3rd International Computer Assisted Assessment Conference. Loughborough University. Loughborough, UK.

Christie, J. R. (2003). Automated essay marking for content- does it work ? In Proceedings of the 7th International Computer Assisted Assessment Conference. Loughborough University. Loughborough, UK.

Dagan I., Glickman O., & Magnini B. (2005). The PASCAL Recognising Textual Entailment Challenge. In Proceedings of the PASCAL Challenges Workshop on Recognizing Textual Entailment.

de Paiva, V. & Holloway King T. (2008). Designing Testsuites for Grammar-based Systems in Applications. In Proceedings of the Coling 2008 Workshop on Gammar Engineering Across Frameworks, pages 49-56.

Foltz, P.W., Laham, D. & Landauer, T.K. (2003). Automated essay scoring. Applications to Educational technology. http://www-psych.nmsu.edu/%7Epfoltz/reprints/Edmedia99.html

Laham, D., Bennett, W., & Derr, M. (2002). Latent Semantic Analysis for career field analysis and information operations. Paper presented at Interservice/Industry, Simulation and Eduation Conference (I/ITSEC), December 2-5, 2002. Orlando, FL.

Larkey, L. S. (1998) Automatic Essay Grading Using Text Categorization Techniques, in Proceedings of SIGIR-98, 21st ACM International Conference on Research and Development in Information Retrieval. ACM Press, 1998.

Leacock, C. & Chodorow, M. (2003) C-rater : Automated Scoring of Short-Answer Questions. Computers and Humanities. Pp. 389-405.

Mason, O. & Grove-Stephenson, I. (2002) Automated free text marking with paperless school. In M. Danson (Ed.), Proceedings of the sixth International Computer Assisted Assessment Conference, Loughborough University, Loughborough, UK.

Ming, Y. Mikhailov A., & Kuan T. L. (2000). Intelligent Essay Marking System. Learners Together, NgeeANN Polytechnic, Singapore.

Mitchell, T. & Russell, T., Broomhead, P. & Aldridge, N. (2002) Towards robust computerised marking of free-text responses. Proceedings of the 6th International Computer Assisted Assessment Conference.

Mohler, M. & Mihalcea R. (2009). Text-to-text Semantic Similarity for Automatic Short Answer Grading. Proceedings of the European Chapter of the Association for Computational Linguistics, Athens, Greece.

Prasad, R. & Sarkar A. (2000). Comparing test-suite based evaluation and corpus-based evaluation of a wide-coverage grammar for English. In LREC-00, Athens.

Rehder, B. Schreiner, M. E. Wolfe, B. W., Laham, D, Landauer, T. K., & Kintsch, W. (1998). Using Latent Semantics Analysis to assess knowledge : Some technical considerations. Discourse Processes, 25, 337-354.

Rosé, C. P., Roque, A., Bhembe, D. & VanLehn, K. (2003) A hybrid text classification approach for analysis of student essays. Proceedings of the HLT-NAACL 03 Workshop on Educational Applications of NLP.

Rudner, L. M., & Liang, T. (2002). Austomated Essay Scoring Using Bayes' Theorem. In Proceedings of the annual meeting of the National Council on Measurement in Eduation.

Siddiqi, R. & Harrison, C. (2008). A systematic approach to the automated marking of short answer questions. Multitopic Conference, 2008. INMIC 2008. IEEE International.

Sukkarieh, J. Z. & Blackmore, J. (2009). c-rater : Automatic Content Scoring for Short Constructed Responses. Proceedings of the 22nd International Conference for the Florida Aritifical Intelligence Research Society, Florida, USA.

Sukkarieh, J. Z., Pulman, S. G. & Raikes, N. (2003). Auto-marking : using computational linguistics to score short, free text responses. Proceedings of the international association of educational assessment, Manchester, UK.

Sukkarieh, J. Z. (2001). Natural Language for Knowledge Representation. Ph.D Thesis. University of Cambridge, Cambridge, England.

Vanderwende, K. and Dolan W. B. (2006). What Syntax can Contribute in the Entailment Task. In MLCM 2005, LNAI 3944, pp. 205-216. J. Quinonero-Candela et al. (eds., Springer-Verlag).

Williams,R. & Dreher, H. (2004). Automatically Grading Essays with Markit©. Proceedings of Informing Science 2004 Conference, Rockhampton, Queensland, Australia, June 25-28, 2004.

Wiemer-Hastings, P., Wiemer-Hastings, K. & Graesser, A. (1999). Improving an intelligent tutor's comprehension of students with Latent Semantic Analysis. In S. P. Lajoie and M. vivet, Artificial Intelligence in Education pp. 535-542. Amsterdam: IOS Press.