Automating Model Building in c-rater

Jana Z. Sukkarieh
Educational Testing Service
Rosedale Road, Princeton, NJ 08541
jsukkarieh@ets.org

Svetlana Stoyanchev
Stony Brook University
Stony Brook, NY, 11794
svetastenchikova@gmail.com

Abstract

c-rater is Educational Testing Service's technology for the content scoring of short student responses. A major step in the scoring process is Model Building where variants of model answers are generated that correspond to the rubric for each item or test question. Until recently, Model Building was knowledge-engineered (KE) and hence labor and time intensive. In this paper, we describe our approach to automating Model Building in c-rater. We show that c-rater achieves comparable accuracy on automatically built and KE models.

1 Introduction

c-rater (Leacock and Chodorow, 2003) is Educational Testing Service's (ETS) technology for the automatic content scoring of short free-text student answers, ranging in length from a few words to approximately 100 words. While other content scoring systems [e.g., Intelligent. Essay Assessor (Foltz, Laham and Landauer, 2003), SEAR (Christie, 1999), IntelliMetric (Vantage Learning Tech, 2000)] take a holistic[1] approach, c-rater takes an analytical approach to scoring content. The item rubrics specify content in terms of main points or concepts required to appear in a student's correct answer. An example of a test question or item follows:

Item 1 (Full credit: 2 points)	*Concepts or main/key points:*
Stimulus: A Reading passage *Prompt:* In the space below, write the question that Alice was most likely trying to answer when she performed Step *B*.	**C_1:** How does rain formation occur in winter? **C_2:** How is rain formed? **C_3:** How do temperature and altitude contribute to the formation of rain?
Scoring rules: 2 points for C1 1 for C2 (only if C1 is not present) 1 for C3 (only if C1 and C2 are not present) Otherwise 0	

We view c-rater's task as a **textual entailment (TE)** problem. We use TE here to mean either a paraphrase or an inference (up to the context of the item or test question). c-rater's task is reduced to a TE problem in the following way:

> **Given** a concept, *C*, (e.g., "body increases its temperature") **and** a student answer, *A*, (e.g., either "the body raises temperature," "the body responded. His temperature was 37◦ and now it is 38◦," or "Max has a fever") **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).

There are four main steps in c-rater. The first one is **Model Building (MB)**, where a set of model answers are generated (either manually or automatically). Second, c-rater automatically processes model answers and students' answers using a set of natural language processing (NLP) tools and extracts the linguistic features. Third, the matching algorithm **Goldmap** uses the linguistic features culminated from both MB and NLP to automatically determine whether a student's response entails the expected concepts. Finally, c-rater applies

[1] Holistic means an overall score is given for a student's answer as opposed to scores for individual components of a student's answer.

Proceedings of the 2009 Workshop on Applied Textual Inference, ACL-IJCNLP 2009, pages 61–69,
Suntec, Singapore, 6 August 2009.

the scoring rules to produce a score and feedback that justifies the score to the student.

Until recently, MB was knowledge-engineered (KE). The KE approach for one item required, on average, 12 hours of time and labor. This paper describes our approach to automatic MB. We show that c-rater achieves comparable accuracy on automatically- and manually-built models. Section 2 outlines others' work in this domain and emphasizes the contribution of this paper. Section 3 outlines c-rater. In Section 4, we describe how MB works. Section 5 explains how we automate the process. Prior to the conclusion, we report the evaluation of this work.

2 Automatic Content Scoring: Others' Work

A few systems that deal with both **short answers** and **analytic-based** content exist. The task, in general, is reduced to comparing a student's answer to a model answer. Recent work by Mohler and Mihalcea (2009) at the University of North Texas uses unsupervised methods in text-to-text semantic similarity comparing unseen students' answers to one correct answer. Previous work, including c-rater, used supervised techniques to compare unseen students' answers to the space of potentially "all possible correct answers" specified in the rubric of the item at hand. The techniques varied from information extraction with knowledge-engineered patterns representing the model answers [Automark at Intelligent Assessment Technologies (Mitchell, 2002), the Oxford-UCLES system (Sukkarieh, et. al., 2003) at the University of Oxford] to data mining techniques using very shallow linguistic features [e.g., Sukkarieh and Pulman (2005) and CarmelTC at Carnegie Mellon University (Rose, et al. 2003)]. Data mining techniques proved not to be very transparent when digging up justifications for scores.

c-rater's model building process is similar to generating patterns but the patterns in c-rater are written in English instead of a formal language. The aim of the process is to produce a non-trivial space of possible correct answers guided by a subset of the students' answers. The motivation is that the best place to look for variations and refinements for the rubric is the students' answers. This is what test developers do before piloting a large-scale exam. From an NLP point of view, the idea is that generating this space will make scoring an unseen answer easier than just having one correct answer. However, similar to what other systems reported, generating manually-engineered patterns is very costly. In Sukkarieh et al. (2004) there was an attempt to generate patterns automatically but the results reported were not comparable to those using manually-generated patterns. This paper presents improvements on previous supervised approaches by automating the process of model-answer building using well-known NLP methods and resources while yielding comparable results to knowledge-engineered methods.

3 c-rater, in Brief

In c-rater, manual MB has its own graphical interface, **Alchemist**. MB uses the NLP tools and **Goldmap** (which reside in the **c-rater Engine**). On the other hand, Goldmap depends on the model generated. The c-rater Engine performs NLP on input text and concept recognition or TE between the input text and each concept (see Figure 1). First, a student answer is processed for **spelling corrections** in an attempt to decrease the noise for subsequent NLP tools. In the next stage, **parts-of-speech tagging** and **parsing** are performed (the OpenNLP parser is used http://opennlp.sourceforge.net). In the third stage, a parse tree is passed through a **feature extractor**. Manually-generated rules extract features from the parse tree. The result is a flat structure representing phrases, predicates, and relationships between predicates and entities. Each phrase is annotated with a label indicating whether it is independent or dependent. Each entity is annotated with a syntactic and semantic role. In the **pronoun resolution** stage, pronouns are resolved to either an entity in the student's answer or the question. Finally, a **morphology analyzer** reduces words to their lemmas.[2] The culmination of the above tools results in a set of linguistic features used by the matching algorithm, Goldmap. In addition to the item-independent linguistic features collected by the NLP tools, Goldmap uses item-dependent features specified in MB to decide whether a student's answer, *A*, and a model

[2] We do not go into detail, assuming that the reader is familiar with the described NLP techniques.

answer match, i.e. that concept *C* represented in the model answer, is entailed by *A*.

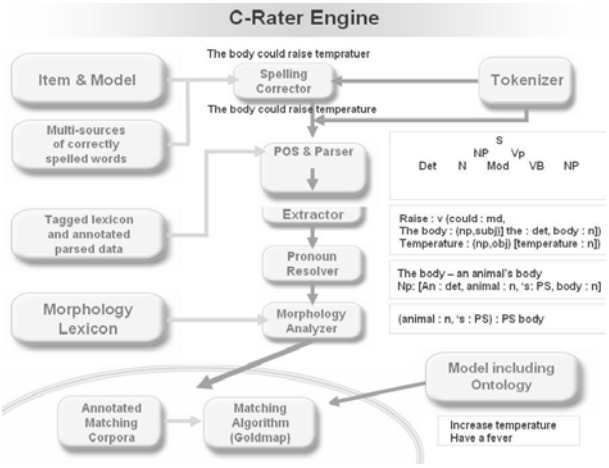

Figure 1. c-rater Engine

4 KE Model Building

A dataset of student answers for an item is split into development (DEV), cross-validation (XVAL), and blind (BLIND) datasets. DEV is used to build the model, XVAL is used to validate it and BLIND is used to evaluate it. All datasets are double-scored holistically by human raters and the scoring process takes an average 3 hours per item for a dataset of roughly 200 answers.

For each concept C_i in item *X*, a model builder uses DEV to create a set of ***Model Sentences*** (MS_{ij}) that s/he believes entails concept C_i in the context of the item. S/he is required to write MS_{ij} in complete sentences. For each model sentence MS_{ij}, the model builder selects the ***Required Lexicon*** (RL_{ijk}), a set of the most essential lexical entities required to appear in a student's answer. Then, for each RL_{ijk}, the model builder selects a set of ***Similar Lexicon*** (SL_{ijkt}), guided by the list of words automatically extracted from a dependency-based thesaurus (cs.ualberta.ca/~lindek/downloads.htm).

The process is exemplified in Figure 2. Presented with the concept, *"What causes rain to form in winter time?,"* a model builder writes model sentences like *"Why does rain fall in the winter?,"* highlights or selects lexical items that s/he believes are the required tokens (e.g., *"why," "rain," "fall," "in," "winter"*) and writes a list of similar lexical entities for each required token if needed (e.g., *{descend, go~down, …}* are similar to words like *"fall"*).[3]

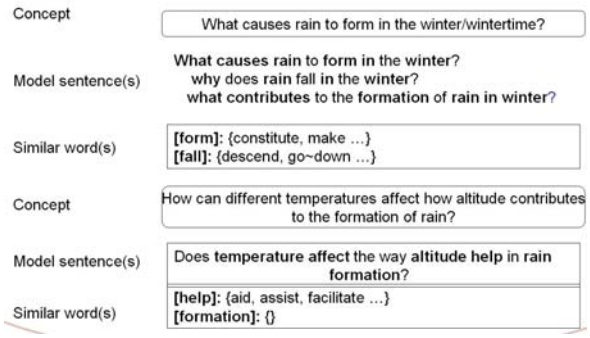

Figure 2. KE Model Building

The model for each item *X* is comprised of the scoring rules**,** the collections of model sentences MS_{ij}, associated lexical entities RL_{ijk}, and corresponding similar lexicon SL_{ijkt}. Each model answer is written in terms of MS_{ij} where:

> MS_{ij} entails C_i for $i=1,\ldots, N$, and N is the number of concepts specified for item *X*. For each concept C_i, Goldmap checks whether answer *A* entails C_i, by checking whether *A* entails one of the model sentences MS_{ij}, given the additional features RL_{ijk} and corresponding SL_{ijkt}.

In practice, model building works as follows. The model builder, guided by the DEV dataset and holistic scores, starts with writing a few model sentences and selects corresponding required (RL_{ijk}) and similar (SL_{ijkt}) lexicon. S/he then uses the **c-rater engine** to automatically evaluate the model using the DEV dataset, i.e., using the model produced up to that point. Goldmap is used to detect if any answers in the DEV dataset contain any of the model sentences and scores are assigned for each answer. If the scoring agreement between c-rater and each of the two human raters (in terms of a kappa statistic) is much lower than that between the two human raters, then the model is judged unsuitable and the process continues iteratively until kappa statistics on the DEV dataset are satisfactory, i.e., c-rater's agreement with human raters is as high as the kappa between human raters. Once kappa statistics on DEV are satisfactory, the model builder uses

[3] We use lexicon, lexical entities, words, terms and tokens interchangeably meaning either uni- or bi-grams.

c-rater to evaluate the model on the XVAL dataset automatically. Again, until the scoring agreement between c-rater and human raters on XVAL dataset is satisfactory, the model builder iteratively changes the model. Unlike the DEV dataset, the XVAL dataset is never seen by a model builder. The logic here is that over-fitting DEV is a concern, making it hard or impossible to generalize beyond this set. Hence, the results on XVAL can help prevent over-fitting and ideally would predict results over unseen data.

Note that a model builder can introduce what we call a negative concept C_i^{-1} for a concept C_i and adjust the scoring rules accordingly. When this happens, a model builder writes model sentences $MS_i^{-1}{}_j$ entailing C_i^{-1} , and selects required words $RL_i^{-1}{}_{jk}$ and corresponding similar words $SL_i^{-1}{}_{jkt}$ in the same way for any other (positive) concept.

On average, MB takes 12 hours of manual work per item (plus 2 hours, on average, for an optional model review by someone other than the model builder). This process is time consuming and error-prone despite utilizing a user-friendly interface like Alchemist. In addition, the satisfaction criterion while building a model is subjective to the model builder.

5 Automated Model Building

The process of writing model sentences described above involves: 1) finding the parts of students' answers containing the concept for each expected concept, 2) abstracting over "similar" parts, and 3) representing the abstraction in one (or more) model sentence(s). The process, as mentioned earlier, is similar to writing rules for information extraction, but here one writes them in English sentences and not in a formal language. In practice, there is no mechanism in Alchemist to cluster "similar" parts and MB, in this aspect, is not performed in any systematic manner. Hence, we introduce what we call **concept-based scoring** – used instead of the holistic human scoring. In concept-based scoring, human raters annotate students' responses for each concept *C*, and highlight the part of the answer that entails *C*. In Sukkarieh and Blackmore (2009), we describe concept-based scoring in detail and how this helps in the KE-MB approach. In this paper, we extend the approach by showing how concept-based scores used in the automated approach reduce the time needed for MB substantially while yielding comparable results. Concept-based scoring is done manually. On average, it takes around 3.5 hours per item for a dataset of roughly 200 answers.

The MB process is reduced to:

1. Concept-based scoring
2. Automatically selecting required lexicon
3. Automatically selecting similar lexicon

While holistic scoring takes on average 3 hours for a dataset of 200 answers, concept-based scoring takes 3.5 hours for the same set. However, automated MB takes 0 hours of human intervention–a substantial reduction over the 12 hours required for manual MB.

5.1 Concept-based Scoring

We have developed a concept-based scoring interface (CBS) that can be customized for each item [due to lack of space we do not include an illustration]. The CBS interface displays a student's answer to an item and all of the concepts corresponding to that item. The terms *{Absent, Present, Negated}* are what we call analytic or concept-based scores. Using CBS, the human scorer clicks *Present* when a concept is present and *Negated* when a concept is negated or refuted (the default is *Absent*). This is done for each concept. The human scorer also highlights the part of a student's answer that entails the concept in the context of the item. We call a quote corresponding to concept *C* 'Positive Evidence' or 'Negative Evidence' for *Present* and *Negated*, respectively. For example, assume a student answer for Item 1 is *"Her research tells us a lot about rain and hail; in particular, the impact that temperature variations have on altitude contribute to the formation of rain."* For **Concept C_3**, the human rater highlights the Positive Evidence, *"the impact that temperature variations have on altitude contribute to the formation of rain."* Parts of answers corresponding to one piece of Evidence (positive or negative) do not need to be in the same sentence and could be scattered over a few lines.

Similar to the KE approach, we split the double-concept-based scored dataset into DEV and XVAL sets. However, the splitting is done

according to the presence (or absence) of a concept. We use stratified sampling (Tucker, 1998) trying to uniformly split data such that each concept is represented in the DEV as well as the XVAL datasets. As mentioned earlier, the KE approach can include negative concepts; currently we do not use Negative Evidence automatically. In the remainder of this paper, Evidence is taken to mean the collection of Positive Evidence.

5.2 Automatically Selecting Model Sentences

Motivation

During manual MB with Alchemist, a model builder is guided by the complete set of students' answers in the DEV dataset, including holistic scores. Concept-based scoring allows a model builder, if we were to continue the manual MB, to be guided by concept-based scores and students' answers highlighted with the Evidence that corresponds to each concept when writing model sentences as shown, where MS_{ij} entails C_i and E_{ir} entails C_i.

Concept C_i	***Evidence E_{ir}***	***MS_{ij}***
C_1	E_{11}	MS_{11}
	E_{1s1}	MS_{1t1}
C_2	E_{21}	MS_{21}
	E_{2s2}	MS_{2t2}
C_n	…	…

Further, students may misspell, write ungrammatically, or use incomplete sentences. Hence, Evidence may contain spelling and grammatical errors. Evidence may also be in the form of incomplete sentences. Although human model builders generating sentences with Alchemist are asked to write complete $MS_{ij,}$, there is no reason why $MS_{ij,}$ needs to be in the form of complete sentences. The NLP tools in the c-rater engine can cope with a reasonable amount of misspelled words as well as ungrammatical and/or incomplete sentences.

We observe the following:

1. Concepts are seen as a set of model sentences that are subsumed by the list of model sentences built by humans

2. Evidence is seen as a list of model "sentences" that nearly subsume the set generated by humans (i.e., the intersection is not empty)

Approach

In the automatic approach, we select the Evidence highlighted in the DEV dataset as MS_{ij}s. We either choose the intersection of Evidence (i.e., where both human raters agree) or the union (i.e., highlighted by either human) as entailing a concept.

5.3 Automatically Selecting Required Lexicon

Motivation

Required lexicon for an item includes the most essential lexicon for this item. In the KE approach, the required lexicon is selected by the model builder, who makes a judgment about it. In Alchemist, a model builder is presented with a tokenized model sentence and s/he clicks on a token to select it as a required lexical entity.

We have observed that selecting required lexicon RL_{ijk} involves ignoring or removing noise, such as stop-words (e.g., "a," "the," "to," etc.), from the presented model sentence. For example, a model builder may select the words, *"how*," *"rain*," *"formation*," and *"winter"* in the model sentence *"How does rain formation occur in the winter?"* and ignore the rest. In addition, there might be words other than stop-words that can be ignored. For example, if a model builder writes, *"It may help Alice and scientists to know **how rain formation** occurs in the **winter**"* – the tokens *"scientists"* and *"Alice"* are not stop-words and can be ignored.

Approach

We evaluate five methods of automatically selecting the required lexicon:

1. Consider all tokens in MS_{ij}
2. Consider all tokens in MS_{ij} without stop-words
3. Consider all heads of NPs and VPs (nouns and verbs)
4. Consider all heads of all various syntactic roles including adjectives and adverbs
5. Consider the lexicon with the highest mutual information measures, with all lexical tokens in model sentences corresponding to the same concept

The first method does not need any elaboration. In the following, we briefly elaborate on each of the other methods.

5.3.1 All Words Without Stop Lexicon

In addition to the list of stop-words provided in Van Rijsbergen's book (Rijsbergen, 2004) and the ones we extracted from WordNet 2.0 http://wordnet.princeton.edu/ (except for "zero," "minus," "plus," and "opposite"), we have developed a list of approximately 2,000 stop-words based on students' data. This includes various interjections and common short message service (SMS) abbreviations that are found in students' data (see Table 1 for examples).

1. Umm	2. Aka	3. Coz
4. Viz.	5. e.g.	6. Hmm
7. Phew	8. Aha	9. Wow
10. Ta	11.Yippee	12. NTHING
13. Dont know	14. Nada	15. Guess
16. Yoink	17. RUOK	18. SPK

Table 1. Student-driven stop-words

5.3.2 Head Words of Noun and Verb Phrases

The feature extractor in c-rater, mentioned in Section 2, labels the various noun and verb phrases with a corresponding syntactic or semantic role using in-house developed rules. We extract the heads of these by simply considering the rightmost lexical entity with an expected POS tag, i.e., for noun phrases we look for the rightmost nominal lexical entity, for verb phrases we look for the rightmost verbs.

5.3.3 Head Words of all Phrases

We consider all phrases or syntactic roles, i.e., not only noun and verb phrases but also adjective and adverb phrases.

5.3.4 Words with Highest Mutual Information

The mutual information (MI) method measures the mutual dependence of two variables. MI in natural language tasks has been used for information retrieval (Manning et. al., 2008) and for feature selection in classification tasks (Stoyanchev and Stent, 2009).

Here, MI selects words that are indicative of the correct answer while filtering out the words that are also frequent in incorrect answers. Our algorithm selects a lexical term if it has high mutual dependence with a *correct concept* or Evidence in students' answers. For each term mentioned in a students' answer we compute mutual information measure (I):

$$I = \frac{N_{11}}{N} * log_2 \frac{N * N_{11}}{N_{1.} * N_{.1}} + \frac{N_{01}}{N} * log_2 \frac{N * N_{01}}{N_{0.} * N_{.1}} + \frac{N_{10}}{N} * log_2 \frac{N * N_{10}}{N_{1.} * N_{.0}} + \frac{N_{00}}{N} * log_2 \frac{N * N_{00}}{N_{0.} * N_{.0}}$$

where N_{11} is the number of student answers with the term co-occurring with a correct concept or Evidence, N_{01} is the number of student answers with a *correct concept* but without the term, N_{10} is the number of student answers with the term but without a *correct concept*, N_{00} is the number of student answers with neither the term nor a correct concept, $N_{1.}$ is the total number of student answers with the term, $N_{.1}$ is the total number of utterances with a *correct concept*, and N is the total number of utterances. The MI method selects the terms or words predictive of both presence and absence of a concept. In this task we are interested in finding the terms that indicate presence of a *correct concept*. We ignore the words that are more likely to occur without the concept (the words for which $N_{11} < N_{10}$). In this study, after looking at the list of words produced, we simply selected the top 40 words with the highest mutual information measure.

5.4 Automatically Selecting Similar Lexicon

Motivation

In the KE approach, once a model builder selects a required word, a screen on Alchemist lists similar words extracted automatically from Dekang Lin's dependency-based thesaurus. The model builder can also use other resources like Roget's thesaurus (http://gutenberg.org/etext/22) and WordNet 3.0 (http://wordnet.princeton.edu/). The model builder can also write her/his own words that s/he believes are similar to the required word.

Approach

Other than choosing no similar lexicon to a required word *W*, automatically selecting simi-

lar lexicon consists of the following experiments:

1. All words similar to *W* in Dekang Lin's generated list
2. Direct synonyms for *W* or its lemma from WordNet 3.0 (excluding compounds). Compounds are excluded because we noticed many irrelevant compounds that could not replace uni-grams in our data.
3. All similar words for *W* or its lemma from WordNet 3.0, i.e., direct synonyms, related words and hypernyms (excluding compounds). Hypernyms of *W* are restricted to a maximum of 2 levels up from *W*

To summarize, for each concept in the KE approach, a model builder writes a set of Model Sentences, manually selects Required Lexicon and Similar Lexicon for each required word. In the automated approach, all of the above is selected automatically. Table 2 summarizes the methods or experiments. We refer to a method or experiment in the order of selection of RL_{ijk} and SL_{ijkt}; e.g., we denote the method where all words were required and similar lexicon chosen from WordNet Direct synonyms by AWD. HSVocWA denotes the method where heads of NPs and VPs with similar words from WordNet All, i.e., direct, related, and hypernyms are selected. A method name preceded by I or U refers to Evidence Intersection or Union, respectively. For each item, there are 40 experiments/methods performed with Evidence as model sentences.

Model Sentences	Required Lexicon	Similar Lexicon
Concepts **(C)**	All words **(A)**	None chosen **(N)**
Evidence Intersection **(I)**	All words with no stopwords **(S)**	Lin all **(L)**
Evidence Union **(U)**	Heads of NPs and VPs **(HSvoc)** Heads of all phrases **(HA)** Highest Mutual information measure **(M)**	WordNet direct synonyms **(WD)** WordNet all similar words **(WA)**

Table 2. Parameters and "Values" of Model Building

Before presenting the evaluation results, we make a note about spelling correction. c-rater has its own automatic spelling corrector. Here, we only outline how spelling correction relates to a model. In the KE approach, model sentences are assumed to not having spelling errors. We use the model sentences, the stimulus (if it exists), and the prompt of the item for additional guidance to select the correctly-spelled word from a list of potential correctly-spelled words designated by the spelling corrector. On the other hand, the Evidence can be misspelled. Consequently, when the Evidence is considered for model sentences, the spelling corrector first performs spelling correction on the Evidence, using stimulus, concepts, and prompts as guides. The students' answers are then corrected, as in the KE approach.

6 Evaluation

The study involves 12 test items developed at ETS for grades 7 and 8. There are seven Reading Comprehension items, denoted R1-R7 and five Mathematics items, denoted M1-M5. Score points for the items range from 0 to 3 and the number of concepts ranges from 2 to 7. The answers for these items were collected in schools in Maine, USA. The number of answers collected for each item ranges from 190-264. Answers were concept-based scored by two human raters (H1, H2). We split the double-scored students' answers available into DEV (90-100 answers), XVAL (40-50) and BLIND (60-114). Training data refer to DEV together with XVAL datasets. Results are reported in terms of un-weighted kappa, representing scoring agreement with humans on the BLIND dataset. H1/2 refers to the agreement between the two humans, c-H1/2 denotes the average of kappa values between c-rater and each human (c-H1 and c-H2). Table 3 reports the best kappa over the 40 experiments on BLIND (Auto I or U). The baseline (Auto C) uses concepts as model sentences.

Item	#Training (Blind)	H1/2	Manual	Auto C	Auto I or U
			c-H1/2	**c-H1/2**	**c-H1/2**
R1	150 (114)	1.0	**0.94**	0.51	**0.97**
R2	150 (113)	0.76	**0.69**	0.28	**0.76**
R3	150 (107)	0.96	**0.87**	0.18	**0.88**
R4	150 (66)	0.77	**0.71**	0.46	**0.75**
R5	130 (60)	0.71	**0.58**	0.22	**0.61**
R6	130 (61)	0.71	**0.73**	0.23	**0.77**
R7	130 (61)	0.87	**0.55**	0.42	**0.42**
M1	130 (67)	0.71	**0.6**	0.0	**0.66**
M2	130 (67)	0.8	**0.71**	0.54	**0.67**
M3	130 (67)	0.86	**0.76**	0.0	**0.79**
M4	130 (67)	0.87	**0.82**	0.13	**0.82**
M5	130 (67)	0.77	**0.63**	0.29	**0.65**

Table 3. Best on BLIND over all experiments

The accuracy using the automated approach with Evidence as model sentences is comparable to that of the KE approach (noted in the column labeled, "Manual") with a 0.1 maximum difference in un-weighted kappa statistics. The first methods (in terms of running order) yielding the best results for the items (in order of appearance in Table 3) are ISWD, ISW, ISN, IMN, IHSVocN, UHALA, ISN, UHSVocN, SLA, ISN, IHAN and IHSVocWA. The methods yielding the best results (regardless of running order) for all items using the Evidence were:

IHAN	U/IHAWD	IHAWA
U/IHALA	U/IHSvocN	IHSvocWA
UHSvocLA	UHSvocWA	UHSvocWD
U/ISLA	U/ISN	U/ISWA
U/ISWD	U/IAWA	IMN
IMWD		

This approach was only evaluated on a small number of items. We expect that some methods will outperform others through additional evaluation.

In an operational setting (i.e., not a research environment), we must choose a model before we score the BLIND data. Hence, a voting strategy over all the experiments has to be devised based on the results on DEV and XVAL. Following our original logic, i.e., using XVAL to avoid over-fitting and predicting the results of BLIND, we implemented a simple voting strategy. We considered c-H1/2 on XVAL for each experiment. We found the maximum over all the c-H1/2 for all experiments. The model corresponding to the maximum was considered the model for the item and used to score the BLIND data. When there was a tie, the first method to yield the maximum *W* chosen. Table 4 shows the results on BLIND using the voting strategy. The results are comparable to those of the manual approach except for R7 which has 7 concepts, the highest number of concepts among all items. The results also show that the voting strategy did not select the "best" model or experiment. We notice that some methods were better in detecting whether an answer entailed a concept *C* than detecting whether it entailed another concept *D*, specified for the same item. This implies that the voting strategy will have to be a function that not only considers the overall kappa agreement (i.e., holistic scores), but concept-based agreement (i.e., using concept-based scores). Next, we noticed that for R7, XVAL did not predict the results on BLIND. This was mainly due to the inability to apply stratified sampling with such a small sample size when there are 7 concepts involved. Further, we may need to take advantage of the training data differently, e.g. an *n*-fold cross-validation approach. Finally, when there is a tie, factors other than running order should be considered.

Item	#Training (Blind)	H1/2	Manual	Auto (C)	Auto (I or U)
			c-H1/2	c-H1/2	c-H1/2
R1	150 (114)	1.0	**0.94**	0.51	**0.88**
R2	150 (113)	0.76	**0.69**	0.18	**0.61**
R3	150 (107)	0.96	**0.87**	0.18	**0.86**
R4	150 (66)	0.77	**0.71**	0.38	**0.67**
R5	130 (60)	0.71	**0.58**	0.17	**0.51**
R6	130 (61)	0.71	**0.73**	0.13	**0.73**
R7	130 (61)	0.87	**0.55**	0.39	**0.16**
M1	130 67)	0.71	**0.6**	0.0	**0.65**
M2	130 67)	0.8	**0.71**	0.54	**0.58**
M3	130 67)	0.86	**0.76**	0.0	**0.79**
M4	130 67)	0.87	**0.82**	0.13	**0.68**
M5	130 67)	0.77	**0.63**	0.26	**0.49**

Table 4. Voting Strategy results on BLIND

In all of the above experiments, the Evidence was corrected using the c-rater's automatic spelling corrector using the stimulus (in case of Reading), the concepts, and the prompts to guide the selection of the correctly-spelled words.

7 Conclusion

Analytic-based content scoring is an application of textual entailment. The complexity of the problem increases due to the noise in student data, the context of an item, and different subject areas. In this paper, we have shown that building a c-rater scoring model for an item can be reduced from 12 to 0 hours of human intervention with comparable scoring performance. This is a significant improvement on research to date using supervised techniques. In addition, as far as we know, no one other than Calvo et al. (2005) made any comparisons between a manually-built "thesaurus" (e.g. WordNet) and an automatically-generated "thesaurus" (e.g. Dekang Lin's database) in an NLP task or application prior to our work. Our next step is to evaluate (and refine) the approach on a larger set of items. Further improvements will include using Negative Evidence, automating concept-based scoring, investigating a context-sensitive selection of similar words using the students' answers and experimenting with various voting strategies. Finally, we need to compare the results reported using unsupervised techniques on the same items and datasets if possible.

Acknowledgments

Special thanks to Michael Flor, Rene Lawless, Sarah Ohls and Waverely VanWinkle.

References

Calvo H., Gelbukh A., and Kilgarriff A. (2005). Distributional thesaurus vs. WordNet: A comparison of backoff techniques for unsupervised PP attachment. In *CICLing*.

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.

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

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

Manning C. D., Raghavan P., and Sch¨utze H. (2008). Introduction to Information Retrieval. Cambridge University Press.

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

Mohler M. and 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, March 2009.

Rosé, C. P. and Roque, A. and Bhembe, D. and 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.

Stoyanchev S. and Stent A. (2009). Predicting Concept Types in User Corrections in Dialog. Proceedings of EACL Workshop on the Semantic Representation of Spoken Language. Athens, Greece.

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

Sukkarieh, J.Z. and Stephen G. Pulman (2005). Information Extraction and Machine Learning: Auto-marking short free-text responses for Science questions. Proceedings of the 12th International conference on Artificial Intelligence in Education, Amsterdam, The Netherlands.

Sukkarieh, J.Z. Pulman S. G. and Raikes, N. (2004). Auto-marking 2: An update on the UCLES-Oxford University research into using computational linguistics to score short, free text responses. Proceedings of the AIEA, Philadelphia, USA.

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

Tucker H. G. (1998) Mathematical Methods in Sample Surveys. Series on multivariate analysis Vol. 3. University of California, Irvine.

Van Rijsbergen C. J. (2004) The Geometry of Information Retrieval. Cambridge University Press. The Edinburgh Building, Cambridge, CB2 2RU, UK.

Vantage. (2000) A study of expert scoring and IntelliMetric scoring accuracy for dimensional scoring of grade 11 student writing responses. Technical report RB-397, Vantage Learning Tech.