Automated scoring of short constructed responses

Suitability of existing systems for usage in University of Helsinki CS1 and CS2 courses

Leo Leppänen

Seminar report

UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, April 20, 2015

Contents

1 Abstract

The ever-increasing pressure to do more and better with less resources has been a driving factor for the creation and implementation of automated tutoring and grading systems. University of Helsinki Department of Computer Science already employs multiple such systems: TestMyCode is an assessment service that provides students with programming exercises, marks their answers and provides feedback based on predefined tests. Other similar systems are TitoTrainer and SQL-Trainer. These systems all share a common characteristic: they take input, complete certain steps and more-or-less blindly compare the output to known correct values.

While automating these kinds of exercises already saves valuable time for the teacher and provides the student with more immediate feedback, it's becoming apparent that more sophisticated exercises - and more sophisticated graders - are needed to further improve the situation.

The department has lately implemented both multiple choice questions and questions requiring short-form free-text answers in the material for its CS1 and CS2 courses. While the multiple question exercises are trivial to grade, no automatic grader for the free-text answers exists as of now. This paper examines multiple such systems used elsewhere and tries to preliminarily assess their suitability for the department's unique needs based on freely available published works.

2 Introduction

As universities face increasing budgetary constraints, the act of balancing budget and quality of teaching becomes more difficult. As a way of both saving money and increasing the quality of teaching, University of Helsinki Department of Computer Science has in recent years automated much of its CS1 and CS2 course work with the Test My Code exercise grading platform

[VVLP13]. Recently, the Test My Code platform has been complemented by

This paper is primarily an overview of currently available grading systems for short-form constructed responses. The paper also attempts to determine the suitability of aforementioned automated graders for grading students' answers to shot-form self-explanation questions.

Several concerns can be identified concerning the suitability of these systems for use with self-explanation questions. Some of them (1-2) concern the language and field of the environment:

1. Whether existing systems are usable with Computer Science as a domain

2. Whether existing systems are usable with the Finnish language

Out of these two, especially number two is of great concern due to the linguistic differences between Finnish and English languages. A more in-depth look at these linguistic differences is found in section 5.2.

Since self-explanation questions are numerous and change often, it is possible that the time and effort of introducing a new question to these grading systems would offset and overshadow the time saved by their implementation. Based on this, further concerns (3-5) are identified:

3. Whether existing systems can be introduced in a sensible time scale.

4. Whether the use of these systems requires considerable amounts of pre-graded training data

5. Whether the introduction of a new question or question type takes too much resources for the systems to be of use.

3 Constructed response questions in University of Helsinki CS1 and CS2 courses

While multiple definitions of the term ”constructed response” exist, the most common is that a constructed response question is any question that requires a free-form answer [Ben91]. This answers can be as short as a single word – in the case of a ”fill in the blank” questions – or as longs as multiple pages. As is perhaps apparent from the openness of the previous definition, the term ”constructed response” is less defined by itself than as an opposite for the term ”chosen response question”, more commonly known as multiple-choice questions.

With such a wide definition, the constructed response is a superset of question types such as essay and mathematical proofs. Instead of observing such a large spread of answer types, this paper focuses on *short, free-text* constructed response questions. Short in this case is defined as averaging one to three sentences. These questions are different from essays in two major ways. Firstly, they are in general shorter. Secondly, they place no emphasis on the style or the grammar of the answer. The correctness is evaluated merely on the factual correctness of the student's statement. Stylistic and grammatical correctness is only relevant tangentially: unreadable and undecipherable answers can not be evaluated on their factual content.

University of Helsinki Department of Computer Science has recently embedded short-form free-text constructed response questions in its CS1 and CS2 course material. An example of such a question can be seen in figure 1.

Most of the exercises follow a pattern similar to the one presented in figure 1: some source code acting as a background for the question followed by the actual question prompt. In most cases the prompt is essentially two question in one: first a questions that relates to the source code provided followed by a second question that is more technical or theoretical in nature.

Alla on kaverisi hahmottelemaa kurssisuoritus-luokan koodia.

```
public class Kurssisuoritus {
  private String kurssikoodi;
  private int arvosana;

  public Kurssisuoritus(String koodi, int tulos) {
    this.kurssikoodi = koodi;
    this.arvosana = tulos;
  }

  public void tulostaSuoritus() {
    System.out.print("Kurssi " + this.kurssikoodi + " ");

    if(this.arvosana > 0) {
      System.out.println("suoritettu arvosanalla " + this.arvosana);
    } else {
      System.out.println("hylatty");
    }
  }
}
```

Koodi on sinänsä ok, mutta kaverisi on hieman hukassa teemaan liittyvistä käsitteistä. Kerro hänelle mikä on konstruktori ja mitä se tekee. Selvennä lisäksi voiko olion sisäistä tilaa muuttaa olion luomisen jälkeen – kerro myös miten jos voi.

Figure 1: An example of a short free-text question.

These two-in-one prompts might appear to be difficult to automatically grade, but existing systems are not foreign to complicated prompts. For example the c-rater [LC03] has been used with prompts as complex as "Compare and contrast what Mama and Walter in *A Raisin in the Sun* believe to be the most important thing in life or what they ”dream” of. Support your choice for each character with dialogue from the excerpt of the play." [LC03]. The author therefore feels that its reasonable to expect these systems to be able to handle even the two-in-one prompts found in the CS1 and CS2 material.

4 Existing machine scoring systems

Existing machine scoring system for short free-text constructed response questions can be classified in two major categories: those that are ”based on algorithmic manipulation of keywords” [BJ10] and those that employ computational linguistics . This paper will use the term ”keyword-based” of the first category and the term ”NLP-based” of the second. Both of these categories can be divided into further subcategories, especially by whether they derive their rules using machine learning or not.

4.1 Keyword-based grading systems

Keyword-based systems are relatively simple in that they do not employ sophisticated natural language processing (NLP) methods. Rather, they depend on manipulating strings and tokens of the text and matching them to certain keywords and regular expressions to determine whether the answer is correct. One such system is OpenMark [But06], utilized by the UK Open University.

[BJ10] compared the OpenMark systems with a regular expression based grader. Using seven different exercises with $n_{min} = 129$ to $n_{max} = 317$ responses, the OpenMark agreement with the expert assigned grades initially varied between 87.9% and 99.5%, while the regular expression based grader initially achieved accuracies between 88.6% and 98.9%. A further test achieved accuracies ranging from 99.3% to 100.0% for OpenMark and 95.5% to 99.7% for regular expressions. This later test, however, used a common set of known answers for both training and testing which raises questions about overfitting the data.

4.2 NLP-based grading systems

NLP-based grading systems employ methods of natural language processing to assign grades to students' answers.

One such systems is AutoMark (described in [MRBA02]), which utilizes "syntactic-semantic templates" to match a student's answer to a known right or wrong answer. The templates consist of sets of roots or lemmas that are connected. An example template could consists of the sets

$$s \in \{\{Earth_n, world_n\},$$
$$\{rotate_v, revolve_v, orbit_v, travel_v, move_v\},$$
$$\{around, round\},$$
$$\{Sun_n\}\}$$

where each set s_i is preceded by set s_{i-1} and followed by set s_{i+1}. For the above template, one matching sentence would be "The Earth rotates around the Sun".

A 2003 study ([MAWB03]) employed the AutoMark system to grade answers of medical students. For the study, the AutoMark system was manually "trained" using $n_{marked} = 50$ hand marked exercises. I a pilot test, the systems achieved an accuracy of around 99% when $n_{items} = 25$ exercises were answered by $n_{students} = 30$ students. In a further test with $n_{items} = 270$ test items and around $n_{students} = 460$ students, some 1.6% of students had their marks later changed by a moderation group due to errors made by AutoMark. This number only includes students who actively asked for moderation so the actual error percentage is likely somewhat higher. The [BJ10] study also assessed what is essentially the AutoMark system[1], reporting accuracies varying between 89.4% and 99.5%.

[1] The system assessed in the paper is called *Intelligent Assessment Technologies FreeText Author* and is a more developed and commercialized version of the AutoMark system.

Another systems that uses a similar template approach is presented in [SPR03]. It reportedly achieved accuracies around 88%.

Some NLP-based grades use Latent Semantic Analysis (LSA) (see f.ex. [LFL98])to determine whether an answer is correct or not. LSA is a statistical method that is based on the idea that the "aggregate of all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of meaning of words and sets of words to each other" ([LFL98]).

In practice, LSA is a refined bag-of-words method and suffers from all the problems of other bag-of-words methods. For example, previous studies have found that an answer which has a bag of words equal to the the bag of words of a correct answer has only a moderately high correlation with the answer being the correct answer ($r = 0.65$ in [She15]). The problem with the bag-of-words approach is clearly demonstrated by considering the sentences "*Brutus killed Caesar*" and "*Caesar killed Brutus*", both of which result in the same bag of words $\{killed, Caesar, Brutus\}$. Perhaps due to these issues, LSA is uncommon in short-answer context and is mostly used in the context of essay grading.

A third kind of NLP-based systems in represented by *c-rater* (See f.ex. [SB09, LC03, She15]). The c-rater system uses an approach that is somewhat similar to that used by AutoMark in that it tries to match the student's answer to a template of a correct answer. The difference is that instead of a list-of-sets type of template like the AutoMark systems, the c-rater instead has only a one "canonical" answer (or multiple canonical answers if the question has multiple correct answers) and the student's answer is processed to a similar canonical form, after which the processed student's answer is compared to the correct answer. This processing includes solving for syntactic variances, pronoun references, morphological variation and the use of synonyms ([LC03]).

C-rater has achieved accuracies of around 84% in large scale tests with $n > 100000$ and $n = 16625$ ([LC03]).

5 Possible suitability problems

As noted at the start of this paper, multiple possible problems can prevent or hinder the usage of existing machine scoring systems as far as University of Helsinki CS1 and CS2 courses are considered. These problems can be generally categorised as domain, language and set-up problems.

5.1 Domain problems

As a relatively young field, Computer Science lacks the kind of uniform Finnish language terminology that is used within other, more mature fields. This is most clear when one considers the commonness of Anglicisms. A student speaking in Finnish might refer use words ”object” or ”objekti”/”objecti” instead of the more canonical Finnish ”olio” when describing Object Oriented Programming. Many terms also have an ”official” Finnish equivalent that in reality is practically never used (F.ex. ”liputtaminen” for ”tagging” or ”puhurointi” for ”uploading”). This essentially means that any systems that processes the word forms of the input document must be prepared to handle a large amount of ”non-standard” terms and possibly even terms from other languages.

Furthermore, Computer Science documents often contain English language terms that are in essence names and therefore have no Finnish equivalent (F.ex. ”HyperText Markup Language”, ”HyperText Transport Protocol”, ”Comma Separated Values”). These can cause further problems for any stemmers and/or lemmatisers that attempt to process the text based on Finnish language conventions and morphological rules. Continuing this trend of mixing languages, a student’s answer can also contain snippets of source code, possibly mixing Finnish language variables and names with English language reserved words.

5.2 Language problems

The stark systematic differences between the English and the Finnish languages present further problems for adapting these existing systems for use in University of Helsinki. The differences that are relevant from this paper's point of view are primarily syntactic[2] and morphological[3].

5.2.1 Problems caused by morphological differences

Finnish as a language is "very inflectional and compound rich" and its "inflectional and derivational morphology is considerably more complex than that of [..] English" ([KLJJ04]). A good example of this high difference in the amount of inflection and derivation can be seen in Figure 2.

English sentence	**Finnish equivalent**
In my house	Talossani
Out of your house	Talostasi
Even to our houses	Taloihimmekin

Figure 2: Examples of differences in the amounts of inflection and derivation between the English and the Finnish languages.

These differences cause problems on two levels. Firstly, matching tokens using regular expressions or similar methods becomes more difficult. This is caused by the fact that the regular expression must on one hand allow a large variety of inflections and derivations that can be used to spell out the correct answer, but must also rule out an equally large set of inflections that

[2]Syntax is "the study of the principles and processes by which sentences are constructed" ([Cho02])

[3]Morphology is the study of "'forms of words' in different uses and construction" ([Mat91])

possibly change the semantics so that the answer is incorrect, even when another inflection of the same stem would be a part of the correct answer.

Furthermore, while stemming (reducing a word to its stem form[4]) works fine with the English language, lemmatization (reducing a word to its ”dictionary form”[5]) has previously been determined as a better method for clustering Finnish language documents [KLJJ04]. This is a problem because to ensure the best results, any algorithms that use stemming as a (pre)processing step should be changed to use lemmatization instead.

5.2.2 Problems caused by syntactic differences

The English and Finnish languages also differ greatly on the syntactic freedom of the speaker. The Finnish word order has considerable amounts of freedom [KK85], especially when contrasted to the English language. This causes problems for any graders that rely on the sentence structure as a conveyor of information. This in turn means that the grader has to either only use what is essentially a bag of words approach, or alternatively must be configured to allow all grammatically correct sentence structures for each variation of the correct answer. The latter method would result in either increased grader complexity – a large up-front cost of ”translating” the grader – or larger set-up effort for the person building the exercises.

5.3 Set-up cost

The set-up cost of a solution comes in three distinct forms of *translation costs*, *per-course costs* and *per-exercise costs*. The term *cost* in this context means not only the monetary costs but also the time and effort required to complete the set-up.

Translation costs are the costs caused by modifying an existing system

[4] F.ex. *edeltäjistään* → **edeltäj*

[5] F.ex. *edeltäjistään* → *edeltäjä*

so that it can handle Finnish language exercises. These are essentially programming related costs. Per-course set-up efforts include setting up any databases and user accounts. These costs are essentially the costs of installing the system. The per-exercise costs are the costs associated with creating a new exercise using the already implemented and installed grading systems.

The total of these costs are effectively $C_t + (n_c \times C_c) + (n_c \times n_e \times C_e)$ where C_t is the cost of the translation phase, C_c is average per-course cost, C_e is the average per-exercise cost , n_c is the number of courses the systems is used on and n_e is the average number of exercises per course.

As large value of C_c are unlikely, this means that C_e quickly becomes the deciding factor in systems usage costs unless C_t is extremely large.

For example, the University of Helsinki combined CS1 and CS2 course material [VL14] contains a total of 28 free-text short answer questions. If we take this number as indicative of the average number of questions per period, this would mean an average of 14 exercises for a normal single period course. The total implementation costs for the first course would then be $C_t + (1 \times C_c) + (1 \times 14 \times C_e)$, with the costs for replacing an existing course's exercises being $n_e \times C_e$, further demonstrating how important a low C_e is.

6 Effects of suitability problems on existing scoring methods

The possible suitability problems mentioned in section 5 have a different effect on NLP-based and Keyword-based grading systems.

6.1 Keyword-based grading systems

Key-word based grading systems suffer mainly from the language problems described in section 5.2. The relatively free syntax of the Finnish language complicates or even prevents any regular expression based analysis beyond unigrams[6], as the word order can be essentially anything. This same freedom of word order also means that any template based matching must implement templates for all possible word orders, increasing the costs of building an exercise many fold.

This difficulty of analysis beyond the unigram (bag-of-words) level essentially forces the analysis to use a bag of words approach, which – as discussed previously – is a sub-optimal strategy.

On the other hand, the domain problems presented in 5.1 are not necessarily a big issue beyond requiring more synonyms to be entered for each term.

This suggests that the one time set-up effort of any Keyword-based grading systems would be relatively small, but the per exercise set-up effort would be longer than with English language exercises. Using the terminology established in section 5.3, this would mean a low C_t but a high C_e.

The resulting accuracy would also be smaller than that achieved on English language exercises if the assumption of unigram-level only processing turns out to be correct.

6.2 NLP-based grading systems

The translation costs (C_t) of NLP based systems are likely higher than those of keyword-based systems, since the costs include replacing the NLP systems of existing solutions with solutions that function with the Finnish language. While such solutions exist[7], actually modifying the existing systems to use them is probably either costly or – in the case of closed source software – impossible. This means that the implementation is either impossible or has a large C_t. This is further complicated by the domain problems as mentioned in section 5.1, since it's unlikely that any existing systems can handle the

[6]Unigrams are single words or items from a text. They are a special case of n-grams, or sequences of n items from a text or document.

[7]See f.ex. [LAH+11, Pir11, LAD+13]

Anglicisms and multilingual sentences likely to occur in the answers.

On the other hand, if it is possible to implement an NLP-based systems, its C_e is likely lower than that of keyword-based system. This means that if the usage of the system would be sufficiently large, the high C_t would be recouped by the lower C_e over the lifespan of the system.

The domain problems effect on the system accuracy relative to that of similar English language systems is hard to estimate. It is however reasonable to expect the accuracy to suffer to some decree due to the domain problems. How the decrease in accuracy compares to that likely suffered by the keyword-based systems can not be estimated without tests. The author's intuition is that in the end, the NLP based methods would have a higher accuracy than the keyword based systems.

7 Summary

Existing free-text short-answer grading systems can be classified as Natural language processing (NLP) based systems and keyword-based systems. Both of these classes have different strong and weak points, both in general and in relation to the Finnish language.

In general, NLP based methods seem to be more accurate than those that are purely keyword-based. On the other hand, they also are more complex and would require more modification before they can be used in conjuncture with the Finnish language. Keyword-based methods seem to require less modification, but would require large per-exercise set-up efforts to reach any reasonable accuracies.

As both NLP and keyword-based methods have scored around the $80-90\%$ mark in previous studies, it is also questionable whether these systems are mature enough for teachers to wholly depend on them for grading purposes. Based on the domain and language problems detailed in section 5, it seems unlikely that the systems could achieve as great accuracies than those that they achieve with the English language.

It's therefore the view of the author that the systems examined here are not yet mature enough for usage as graders for purposes of determining students' course grades as they are now. It seems more responsible to first test these systems out with a pilot program in a more forgiving environment, perhaps by using the alongside human experts to improve the technology and seek solutions that work especially with the Finnish language.

Such a pilot program should likely start by evaluating keyword-based methods, since they lack the high translation costs of NLP-based methods. While it is unlikely that these keyword-based methods would turn out to be sufficiently accurate, they would likely still provide valuable insights for translation of the NLP methods.

References

[Ben91] Bennett, Randy Elliot: *On the meanings of constructed response.* ETS Research Report Series, 1991(2):i–46, 1991.

[BJ10] Butcher, Philip G and Jordan, Sally E: *A comparison of human and computer marking of short free-text student responses.* Computers & Education, 55(2):489–499, 2010.

[But06] Butcher, PG: *OpenMark examples.* `http://www.open.ac.uk/openmarkexamples/`, 2006. Accessed: 2015-03-19.

[Cho02] Chomsky, Noam: *Syntactic structures.* Walter de Gruyter, 2002.

[KK85] Karttunen, Lauri and Kay, Martin: *Parsing in a free word order language.* Natural language parsing, pages 279–306, 1985.

[KLJJ04] Korenius, Tuomo, Laurikkala, Jorma, Järvelin, Kalervo, and Juhola, Martti: *Stemming and lemmatization in the clustering of Finnish text documents.* In *Proceedings of the thirteenth*

ACM international conference on Information and knowledge management, pages 625–633. ACM, 2004.

[LAD+13] Lindén, Krister, Axelson, Erik, Drobac, Senka, Hardwick, Sam, Kuokkala, Juha, Niemi, Jyrki, Pirinen, Tommi A, and Silfverberg, Miikka: *HFST—a system for creating NLP tools.* In *Systems and Frameworks for Computational Morphology*, pages 53–71. Springer, 2013.

[LAH+11] Lindén, Krister, Axelson, Erik, Hardwick, Sam, Pirinen, Tommi A, and Silfverberg, Miikka: *HFST—framework for compiling and applying morphologies.* In *Systems and Frameworks for Computational Morphology*, pages 67–85. Springer, 2011.

[LC03] Leacock, Claudia and Chodorow, Martin: *C-rater: Automated scoring of short-answer questions.* Computers and the Humanities, 37(4):389–405, 2003.

[LFL98] Landauer, Thomas K, Foltz, Peter W, and Laham, Darrell: *An introduction to latent semantic analysis.* Discourse processes, 25(2-3):259–284, 1998.

[Mat91] Matthews, Peter H: *Morphology (cambridge textbooks in linguistics)*, 1991.

[MAWB03] Mitchell, Tom, Aldridge, Nicola, Williamson, Walter, and Broomhead, Peter: *Computer based testing of medical knowledge.* In *Proceedings of the 7th computer assisted assessment conference*, pages 249–267, 2003.

[MRBA02] Mitchell, Tom, Russell, Terry, Broomhead, Peter, and Aldridge, Nicola: *Towards robust computerised marking of free-text responses.* 2002.

[Pir11] Pirinen, Tommi A: *Modularisation of finnish finite-state language description—towards wide collaboration in open source development of a morphological analyser.* In *Proceedings of Nodalida*, volume 18, pages 299–302. Citeseer, 2011.

[SB09] Sukkarieh, Jana Zuheir and Blackmore, John: *c-rater: Automatic Content Scoring for Short Constructed Responses.* In *FLAIRS Conference*, 2009.

[She15] Shermis, Mark D: *Contrasting state-of-the-art in the machine scoring of short-form constructed responses.* Educational Assessment, 20(1):46–65, 2015.

[SPR03] Sukkarieh, Jana Z, Pulman, Stephen G, and Raikes, Nicholas: *Auto-marking: using computational linguistics to score short, free text responses.* In *th annual conference of the International Association for Educational Assessment (IAEA), Manchester, UK*, 2003.

[VL14] Vihavainen, Arto and Luukkainen, Matti: *Ohjelmoinnin perus- ja jatkokurssin materiaali.* `http://www.cs.helsinki.fi/group/java/s14-materiaali/viikko1/`, 2014.

[VVLP13] Vihavainen, Arto, Vikberg, Thomas, Luukkainen, Matti, and Pärtel, Martin: *Scaffolding students' learning using test my code.* In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pages 117–122. ACM, 2013.