Computer-Aided Generation of Multiple-Choice Tests

Ruslan Mitkov, Le An Ha
School of Humanities, Languages and Social Sciences
University of Wolverhampton, WV1 1SB
Email {r.mitkov, l.a.ha}@wlv.ac.uk

Abstract

This paper describes a novel computer-aided procedure for generating multiple-choice tests from electronic instructional documents. In addition to employing various NLP techniques including term extraction and shallow parsing, the program makes use of language resources such as a corpus and WordNet. The system generates test questions and distractors, offering the user the option to post-edit the test items.

1. Introduction

Multiple-choice tests have proved to be an efficient tool for measuring students' achievement.[1] The manual construction of such tests, however, is a time-consuming and labour-intensive task.

In this paper we seek to provide an alternative to the lengthy and demanding activity of developing multiple-choice tests and propose a new, NLP-based approach for generating tests from narrative texts (textbooks, encyclopaedias). The approach uses a simple set of transformational rules, a shallow parser, automatic term extraction, word sense disambiguation, a corpus and WordNet. While in the current experiment we have used an electronic textbook in linguistics to automatically generate test items in this area, we should note that the methodology is general and can be extended to practically any other area.

To the best of our knowledge, no related work has been reported addressing such a type of application.[2]

[1] This work is not concerned with (and does not discuss) the issue of whether multiple-choice tests are better assessment methodology that other types of tests. What it focuses on is a new NLP methodology to generate multiple-choice tests about facts explicitly stated in a text.

[2] Fairon (1999) reports that their exercises 'can take the appearance of a multiple choice test' (if distractors are added), but does not explain exactly as to how this can be done.

2. NLP-based methodology for generation of multiple-choice test items

The proposed methodology for generating multiple-choice test items is based on the premise that questions should focus on key concepts rather than addressing less central and even irrelevant concepts or ideas. Therefore the first stage of the procedure is to identify domain-specific terms which serve as 'anchors' of each question. By way of example, *syntax* is a prime candidate for a domain-specific term in the sentence "Syntax is the branch of linguistics which studies the way words are put together into sentences". This sentence can be then transformed into questions asking about this term such as "Which branch of linguistics studies the way words are put together into sentences?" or "Which discipline studies the way words are put together into sentences?" both of which can act as stems in multiple-choice test items.

Another important premise is that distractors[3] should be as semantically close to the correct answer as possible so that no additional clues are provided for the students. Semantically close distractors are more plausible and therefore better at distinguishing good, confident students from poor and uncertain ones. In the above example, the distractors for the correct answer *syntax* should preferably be *semantics* or *pragmatics* and not *chemistry* or *football*, for instance.

In order to keep the test item comprehensible and avoid additional complexity, the test questions are generated from declarative sentences using simple transformational rules which, in turn, results in only minimal change of the original wording.

Underpinned by the above principles, a system for computer-aided generation of multiple-choice test items from instructional documents in electronic form has been implemented. The system is built on separate components, which perform the following tasks: (i) term extraction, (ii) selection of distractors and (iii) question generation.

[3] Known also as 'distracters' in the literature of classical test theory.

2.1 Term extraction

To retrieve terms, nouns and noun phrases are first identified, using the FDG shallow parser (Tapanainen and Järvinen 1997). Next, their frequency is counted and sorted, and nouns with a frequency over a certain threshold[4] are considered as *key terms*. In addition, noun phrases having these key terms as heads, and satisfying the regular expression [AN]+N or [AN]*NP[AN]*N (Justeson and Katz 1996), are considered as *terms*. Although this method is very simple,[5] the results show that, for this particular application, the performance is more than acceptable (only 3 questions did not address a domain-specific term). One of the main reasons not to employ more complicated methods for term extraction derives from the small size of the corpus used in the current experiment (10 000 words).

It should be noted that, from a keyword, as in the case of the keyword "*phrase",* a list of semantically close terms including *noun phrase*, *verb phrase*, *adjective phrase* and *adverb phrase* can be obtained. In addition, a word sense disambiguation program is used to identify the correct sense of the alternatives given that WordNet frequently returns an unnecessarily high number of senses. The word sense disambiguation algorithm compares the definition of sense (as extracted from WordNet) and the context of the keyword (words around the keyword in the corpus).

As an illustration, in the following extract (Kies 2003)

> (1) A prepositional phrase at the beginning of a sentence constitutes an introductory modifier.

one of the terms identified is *introductory modifier* which can serve as an 'anchor' for generating the test question.

2.2 Selection of distractors

WordNet is consulted to compute concepts semantically close to the correct answer/concept which can then be selected as distractors. WordNet retrieves hypernyms, hyponyms, and coordinates of the term, if applicable. If WordNet returns too many concepts, those appearing in the corpus are given preference. If, as in (1), the term is a noun phrase and WordNet fails to return any semantically close concept, the corpus is searched for noun phrases with the same head which are then used as distractors.[6] As an illustration, the electronic textbook contains the following noun phrases with *modifier* as the head, each one of which can act as a distractor: *modifier that accompanies a noun*, *associated modifier*, *misplaced modifier*. As a result, the program generates the following multiple-choice test item:

> (2) What does a prepositional phrase at the beginning of a sentence constitute?
>
> i. a modifier that accompanies a noun
> ii. an associated modifier
> iii. an introductory modifier
> iv. a misplaced modifier

2.3 Generation of test questions

Sentences eligible for question generation are those containing domain-specific terms. Another condition for a sentence to be eligible is that its structure is of SVO or SV type.[7] Currently, a number of simple question generation rules have been implemented. Example rules include the transformation of an SVO sentence in which the subject is a term, into the question "Which HVO" where H is a hypernym of the term. Such a rule would generate the question "Which part of speech is the most central element in a clause" from the sentence "The verb is the most central element in a clause". This rule operates in several variants, one being that if the hypernym is a key term, then a 'Which kind of' question may be generated (e.g. 'Transitive verbs require objects' would trigger the question "Which kind of verbs require objects?"). Another rule often used transforms an SVO sentence with object representing a term into the question "What do/does/did the S V". By way of example, this rule would convert the sentence in example (1) into the question "What does a prepositional phrase at the beginning of a sentence constitute?"

The system makes use of agreement rules which ensure the grammaticality of the question generated. These rules also check for agreement between concepts mentioned in the question and the distractors. As an illustration, in addition to the local agreement in the question "What kind of phrases can act as adjectives,

---

[4] For this particular project the threshold has been determined through experiments. The value of the threshold of course depends on a number of parameters such as the size of the corpus, number of nouns etc.

[5] We experimented with the tf.idf method for key term extraction and noted that while precision is slightly higher, recall is much lower. As the time needed to validate a question is much less than the time needed to produce it, we believe that the recall rate is more important.

[6] In the rare case of the program not being able to extract suitable distractors from WordNet or/and from the corpus, no test item is generated.

[7] Sentences of such types are identified by the FDG parser which returns syntax functions.

adverbs and nouns", the alternatives selected will be plural (e.g. *infinitive phrases*, *prepositional phrases*, *adverbial phrases*, *noun phrases*). On the other hand, the alternatives belonging to the test item featuring the question "What grammatical category does a prepositional phrase at the beginning of a sentence constitute?" will be singular.

The generation strategy of multiple-choice items included additional genre-specific heuristics such as discounting examples for further processing, excluding sentences that refer to tables or previously mentioned entities, not splitting compound verbs, etc.

3. In-class experiments and system interface

We introduced a controlled set[8] of the generated test items into a classroom environment in order to obtain sufficient evaluation data related to their acceptability/revision and quality. The controlled set currently consists of 24 test items generated with the help of the program and 12 items produced manually.

A total of 45 undergraduate students in language/linguistics took the class test. The majority of students were from our university, but several students were studying in other UK or European Universities. Students were asked not to spend more than 2 minutes on a test question.

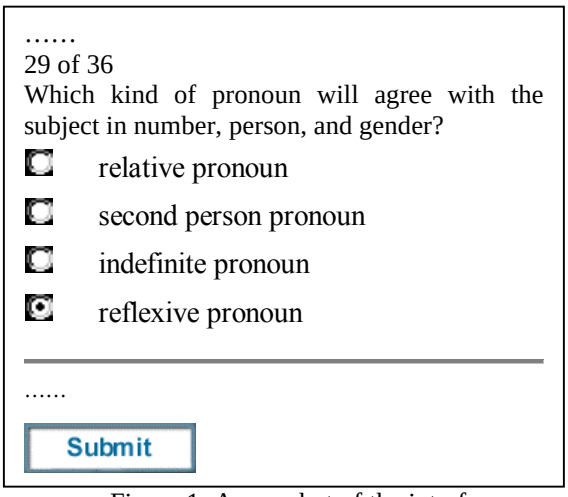

Figure 1: A snapshot of the interface

The system works through the Questionmark Perception web-based testing software which in addition to providing a user-friendly interface, computes diverse statistics related to the test questions answered. Figure 1 shows the interface of the system in a class test environment. The test item displayed is one of the 24 items generated with the help of the system that are used in the experiment.[9]

The current experimental setting does not look at the problem of delivering a balanced test of preset overall difficulty based on random (or constraint-driven) selection of test items. Instead, it focuses on exploring the feasibility of the computer-aided procedure and on the quality of the test items produced.

4. Evaluation

In order to validate the efficiency of the method, we evaluated the performance of the system in two different ways. Firstly, we investigated the efficiency of the procedure by measuring the average time needed to produce a test item with the help of the program as opposed to the average time needed to produce a test item manually.[10] Secondly, we examined the quality of the items generated with the help of the program, and compared it with the quality of the items produced manually. The quality was assessed via standard test theory measures such as discriminating power and difficulty of each test item, and the usefulness of each alternative was applied.

4.1 The procedure of generating test items with the help of the program and its efficiency

The first step of the procedure consists of the automatic generation of test items. The items so generated were then either (i) declared as 'worthy' and accepted for direct use without any revision, or further post-edited before being put into use, or (ii) declared as 'unworthy' and discarded. 'Unworthy' items were those that did not focus on a central concept or required too much revision, and so they were rejected.

The items selected for further post-editing required minor, fair or major revisions. 'Minor' revision describes minor syntactical post-editing of the test question, including minor operations such insertions of articles, correction of spelling and punctuation. 'Fair' revision refers to some grammatical post-editing of the test question, including re-ordering or deletion of words and replacement of one distractor at most. 'Major' revision applied to the generated test items involved more substantial grammatical revision of the test question and replacement of two or more of the

[8] Only items approved by a linguistics lecturer were used in the experiment (e.g. it was made sure that the items addressed material covered by undergraduate students).

[9] The position of the correct answer (in this case 'reflexive pronoun') is generated randomly.

[10] Two graduate students in linguistics acted as post-editors. The same students were involved in the production of test items manually. The texts used were selected with care so that possible influence of potentially similar or familiar texts was minimised. See also the discussion in section 5 on the effect of familiarity.

distractors. As an illustration, the automatically generated test item

> (3) Which kind of language unit seem to be the most obvious component of language, and any theory that fails to account for the contribution of words to the functioning of language is unworthy of our attention?
>
> (a) word
> (b) name
> (c) syllable
> (d) morpheme

was not acceptable in this form and required the deletion of the text 'and any theory that fails to account for the contribution of words to the functioning of language is unworthy of our attention' which was classed as 'fair' revision.

From a total of about 575 items automatically generated by the program, 57% were deemed to be 'worthy' i.e. considered for further use. From the worthy items, 6% were approved for direct class test use without any post-editing and 94% were subjected to post-editing. From the items selected for revision, 17% needed minor revision, 36% needed fair revision and 47% needed major revision.

The time needed to produce 300 test items with the help of the program, including the time necessary to reject items, accept items for further editing or approve for direct use, amounted to 9 hours. The time needed to manually produce 65 questions was 7 hours and 30 minutes. This results in an average of 1 minute and 48 seconds to produce a test item with the help of the program and an average of 6 minutes and 55 seconds to develop a test item manually (Table 1).

	items produced	Time	average time per item
computer-aided	300	540'	1' 48"
Manual	65	450'	6' 55"

Table 1: Effectiveness of the method.

4.2 Analysis of the items generated with the help of the program

Item analysis is an important procedure in classical test theory which provides information as to how well each item has functioned. The item analysis for multiple-choice tests usually consists of the following information (Gronlund 1982): (i) the difficulty of the item, (ii) the discriminating power and (iii) the usefulness[11] of each alternative. This information can tell us if a specific test item was too easy or too hard, how well it discriminated between high and low scorers on the test and whether all of the alternatives functioned as intended. Such types of analysis help improve test items or discard defective items.

In order to conduct this type of analysis, we used a simplified procedure, described in (Gronlund 1982). We arranged the test papers in order from the highest score to the lowest score. We selected one third of the papers and called this the upper group (15 papers). We also selected the same number of papers with the lowest scores and called this the lower group (15 papers). For each item, we counted the number of students in the upper group who selected each alternative; we made the same count for the lower group.

(i) Item Difficulty

We estimated the *Item Difficulty* (ID) by establishing the percentage of students from the two groups who answered the item correctly (ID = C/T x 100, where C is the number who answered the item correctly and T is the total number of students who attempted the item). From the 24 items subjected to analysis, there were 0 *too difficult* and 3 *too easy* items.[12] The *average item difficulty* was 0.75.

(ii) Discriminating Power

We estimated the item's *Discriminating Power* (DP) by comparing the number students in the upper and lower groups who answered the item correctly. It is desirable that the discrimination is *positive* which means that the item differentiates between students in the same way that the total test score does.[13] The formula for computing the *Discriminating Power* is as follows: DP = $(C_U - C_L)$: T/2 where C_U is the number of students in the upper group who answered the item correctly and C_L - the number of the students in the lower group that

[11] Originally called 'effectiveness'. We chose to term this type of analysis 'usefulness' to distinguish it from the (cost/time) 'effectiveness' of the (semi-) automatic procedure as opposed to the manual construction of tests.

[12] For experimental purposes, we consider an item to be 'too difficult' if ID ≤ 0.15 and an item 'too easy' if ID ≥ 0.85.

[13] Zero DP is obtained when an equal number of students in each group respond to the item correctly. On the other hand, negative DP is obtained when more students in the lower group than the upper group answer correctly. Items with zero or negative DP should be either discarded or improved.

	item difficulty			item discriminating power		usefulness of distractors			
	avg item difficulty	too easy	Too difficult	average discriminating power	negative discriminating power	poor	not useful	Total	avg difference
computer-aided	0.75	3	0	0.4	1	6	3	65	1.92
manual	0.59	1	0	0.25	2	10	2	33	1.18

Table 2: Item analysis

did so. Here again T is the total number of students included in the item analysis.[14] The *average DP* for the set of items used in the class test was 0.40. From the analysed test items, there were was only one item that had a *negative discrimination*.

(iii) Usefulness of the distractors

The *usefulness of the distractors* is estimated by comparing the number of students in the upper and lower groups who selected each incorrect alternative. A good distractor should attract more students from the lower group than the upper group.

The evaluation of the distractors estimated the *average difference between students in the lower and upper groups* to be 1.92. Distractors classed as *poor* are those that attract more students from the upper group than from the lower group, and there were 6 such distractors. On the other hand, we term distractors *not useful* if they are selected by no student. The evaluation showed that there were 3 distractors deemed *not useful*.

4.3 Analysis of the items constructed manually

An experiment worthwhile pursing was to conduct item analysis of the manually produced test items and compare the results obtained regarding the items produced with the help of the program. A set of 12 manually produced items were subjected to the above three types of item analysis. There were 0 *too difficult* and 1 *too easy* items. The average *item difficulty* of the items was 0.59. The average *discriminating power* was assessed to be 0.25 and there were 2 items with *negative discrimination*. The evaluation of the *usefulness of the distractors* resulted in an *average difference between students in the upper and lower groups* of 1.18. There were 10 distractors that attracted more students from the upper group and were therefore, declared as *poor* and 2 distractors not selected at all, and therefore deemed to be *not useful*.

Table 2 summarises the item analysis results for both test items produced with the help of the program and those produced by hand.

5. Discussion and plans for future work

The evaluation results clearly show that the construction of multiple-choice test items with the help of the program is much more effective than purely manual construction. We believe that this is the main advantage of the proposed methodology. As an illustration, the development of a test databank of considerable size consisting of 1000 items would require 30 hours of human input when using the program, and 115 hours if done manually. This has direct financial implications as the time and cost in developing test items would be dramatically cut.

At the same time, the test item analysis shows that the quality of test items produced with the help program is not compromised in exchange for time and labour savings. The test items produced with of the program were evaluated as being of very satisfactory quality. As a matter of fact, in many cases they scored even better than those manually produced. Whereas the *item difficulty* factor assessed for manual items emerges as better[15], of those produced with the help of the program, there were only 3 *too easy* items and 0 *too difficult* ones. In addition, whilst the values obtained for the *discriminating power* are not as high as we would have desired, the items produced with the help of the program scored much better on that measure and what is also very important, is that there was only one item among them with *negative discrimination* (as opposed to 2 from those manually constructed). Finally, the analysis of the distractors confirms that it is not possible to class the manually produced test items as better quality than the ones produced with the help of the program. The test items generated with the help of the program scored

[14] Maximum positive DP is obtained only when all students in the upper group answer correctly and no one in the lower group does. An item that has a maximum DP (1.0) would have an ID 0.5; therefore, test authors are advised to construct items at the 0.5 level of difficulty.

[15] Ideally, *item difficulty* should be around the mark of 0.5

better on the number of distractors deemed as *not useful*, were assessed to contain fewer *poor distractors* and had a higher *average difference* between students in the lower and upper groups.

In order to ensure a more objective assessment of the efficiency of the procedure, we plan to run the following experiment. At least 6 months after a specific set of items has been produced with the help of the program, the post-editors involved will be asked to produce another, based on the same material, manually. Similarly, after such a period items originally produced manually will be produced by the same post-editors with the help of the program. Such an experiment is expected to extinguish any effect of familiarity and to provide a more objective measure as to how computer-aided construction of tests is more effective than manual production.

It should be noted that the post-editors were not professional test developers. It would be interesting to investigate the impact of the program on professional test developers. This is an experiment envisaged as part of our future work.

In addition to extending the set of test items to be evaluated and the samples of students taking the test, further work includes experimenting with more sophisticated term extraction techniques and with other more elaborate models for measuring semantic similarity of concepts. We would like to test the feasibility of using collocations from an appropriate domain corpus with a view to extending the choice of plausible distractors. We also envisage the development of a more comprehensive grammar for generating questions, which in turn will involve studying and experimenting with existing question generation theories. As our main objective has been to investigate the feasibility of the methodology, we have so far refrained from more advanced NLP processing of the original documents such as performing anaphora resolution and temporal or spatial reasoning which will certainly allow for more questions to be generated. Future work also envisages evaluation as to what extent the questions cover the course material. Finally, even though the agreement between post-editors appears to be a complex issue, we would like to investigate it in more depth. This agreement should be measured on semantic rather than syntactic principles, as the post-editors may produce syntactically different test questions which are semantically equivalent. Similarly, different distractors may be equally good if they are equal in terms of semantic distance to the correct answer.

6. Conclusion

This paper describes a novel NLP-based and computer-aided procedure for the construction of multiple-choice tests from instructional documents in electronic form. The results from the evaluation conducted suggest that the new procedure is very effective in terms of time and labour, and that the test items produced with the help of the program are not of inferior quality to those produced manually.

References

Fairon, C. (1999). "A Web-based System for Automatic Language Skill Assessment: EVALING". *Proceedings of Computer Mediated Language Assessment and Evaluation in Natural Language Processing Workshop.*

Gronlund, N. (1982) *Constructing achievement tests*. New York: Prentice-Hall Inc.

Justeson, J. S. and S. L. Katz (1996) "Technical terminology: some linguistic properties and an algorithm for identification in text". *Natural Language Engineering*, 3, (2), 259-289.

Kies, D. (2003) *Modern English Grammar*. Online textbook. http://www.papyr.com/hypertextbooks/engl_126/book126.htm

Tapanainen, P. and Järvinen, T. (1997) "A non-projective dependency parser". *Proceedings of the 5th Conference of Applied Natural Language Processing (ANLP-5)*, 64-71.