Natural Language Engineering **1** (1): 1–17. Printed in the United Kingdom

A computer-aided environment for generating multiple-choice test items

RUSLAN MITKOV

LE AN HA

NIKIFOROS KARAMANIS

Research Group in Computational Linguistics
School of Humanities, Languages and Social Sciences
University of Wolverhampton, WV1 1SB.
{r.mitkov, l.a.ha, n.karamanis}@wlv.ac.uk

(*Received 01 May 2005; revised 25 November 2005*)

Abstract

This paper describes a novel computer-aided procedure for generating multiple-choice test items from electronic documents. In addition to employing various Natural Language Processing techniques, including shallow parsing, automatic term extraction, sentence transformation and computing of semantic distance, the system makes use of language resources such as corpora and ontologies. It identifies important concepts in the text and generates questions about these concepts as well as multiple-choice distractors, offering the user the option to post-edit the test items by means of a user-friendly interface. In assisting test developers to produce items in a fast and expedient manner without compromising quality, the tool saves both time and production costs.

1 Introduction

Multiple-choice tests are sets of test items each of which consists of a question or stem (e.g. "Who was voted the best international footballer for 2004?"), the correct answer ("Ronaldinho"), and distractors[1] (such as "Henry", "Ronaldo", "Beckham"). This type of test item has proved to be an efficient tool for measuring test takers' achievement and is used worldwide on a daily basis both for assessment and diagnostics.[2] According to Questionmark Computing Ltd (p.c.), who have licensed their Perception software to approximately three million users so far, 95% of their users employ this software to administrate multiple-choice tests. Despite their popularity, the manual construction of such test items remains a time-consuming

[1] Also known as "distracters" in the literature of classical test theory.

[2] This paper does not discuss whether multiple-choice tests are better for assessment than other types of tests. What we focus on is a new NLP-based methodology to generate multiple-choice test items about facts explicitly stated in single declarative sentences. This methodology does not extend to generating items whose answers require reasoning as this task is beyond the capabilities of most current operational NLP systems.

and labour-intensive task. One of the main challenges in constructing a multiple-choice test item is the selection of plausible distractors which will better distinguish confident test takers from unconfident ones.

This paper proposes an alternative to the lengthy and demanding activity of developing multiple-choice test items manually and provides a new NLP-based methodology for generating such items semi-automatically from an electronic text. This methodology makes use of techniques such as shallow parsing, term extraction, sentence transformation and computation of semantic distance and employs corpora and ontologies such as WordNet. While in the current experiments a textbook in Linguistics was used to automatically generate items in this area, we should note that the methodology is general and can be extended to other fields too.

The initial results of this project were reported in Mitkov and Ha (2003). This paper additionally reports on the post-editing environment we utilised and discusses the results of the error analysis so far. We also present further empirical results including a more extensive evaluation of the quality of the test items produced and the premise that the best test items are those which have distractors as semantically close as possible to the correct answer.

1.1 Related work

The computer-aided generation of test items is a novel research area with just a handful of work reported so far. Fairon (1999) used manually compiled linguistic patterns and manual specification of the distractors to produce exercises which test certain language phenomena and can be either in the form of a multiple-choice question (MCQ) or a fill-in-the-blank item (FBI). More recently, Liu *et al.* (2005) employed techniques for word sense disambiguation to retrieve sentences from a corpus in which the answer carries a specific sense and applied a collocation-based method for selecting distractors. In their setting, test administrators preselect the answer (as well as its part-of-speech and sense) for a specified number of FBIs (or cloze items) aimed at testing foreign learners' proficiency of English. By contrast, our system automatically detects the important concepts in the source text and produces interrogative stems for items which test factual knowledge explicitly conveyed in the text.

Brown *et al.* (2005) automatically generate test items for vocabulary assessment, which may appear in a multiple-choice format. In order to test a student's knowledge of a preselected input word, they produce a non-interrogative stem on the basis of the word's definition or an example of its use in WordNet. Our approach is different in that we do not rely on WordNet for the production of the stem (which always features as a question) but employ it for the selection of distractors.

Hoshino and Nakagawa (2005) discuss how standard classification methods can be used to decide the position of the gap in an FBI. Sumita *et al.* (2005) create FBIs simply by replacing verbs with gaps in an input sentence. Possible distractors are retrieved from a thesaurus and a new sentence is created by placing each distractor in the gap position which is then used as the input for a Web search. If the sentence is found on the Web, the distractor is rendered invalid. This approach is evaluated by showing that estimating the English proficiency of non-native speakers

who took a test consisting of automatically generated items using Item Response Theory (IRT) correlates well with their scores in the Test of English for International Communication (TOEIC). In this paper, we use a standard method for item analysis from classical test theory (Gronlund 1982) which can be applied to evaluate the performance of students whose factual knowledge is unknown before they have taken the test. This is also the first work in which the automatically generated items have been systematically post-edited in an attempt to provide feedback for the further development of the system.

2 Methodology and premises

The proposed methodology for generating multiple-choice test items is based on the premise that test items should focus on key concepts rather than addressing less central or irrelevant notions. Therefore, the first stage of the procedure is to identify domain-specific terms which will serve as the answers for the items. For instance, given the sentence "Syntax studies the way words are put together into sentences", "syntax" is a prime candidate for a domain-specific term. This sentence may then be transformed into the following stem: "Which discipline studies the way words are put together into sentences?"

Another premise is that distractors should be as semantically close as possible to the answer. We shall refer to this premise as the "distractor selection premise". Semantically close distractors are more plausible and therefore better for distinguishing confident test takers from uncertain ones. In the above example, the distractors for the correct answer "syntax" should preferably be "semantics" or "pragmatics" instead of e.g. "chemistry" or "mathematics".

In order to keep the test item comprehensible and avoid additional complexity, a stem is generated from a declarative finite clause using simple transformational rules which result in only minimal changes of the original wording. We shall refer to the clause that gives rise to the stem of a test item as the "source clause".

Underpinned by the above principles, a system for computer-aided generation of multiple-choice test items was implemented using chapters from an online textbook in Linguistics (Kies 2003) as its inputs while the whole textbook served as a domain-specific corpus. The system is built on separate components, which perform the following tasks: (i) term extraction, (ii) stem generation and (iii) distractor selection. Each task is explained in the following sections.

2.1 Term extraction

To retrieve terms from the input chapter, nouns and NPs are first identified using the FDG parser (Tapanainen and Jrvinen 1995). Next, their frequency is counted, and nouns with a frequency over a certain threshold are considered to be key terms. The threshold was determined empirically and depends on several parameters such as the length of the text, the number of nouns it contains, etc.

In addition, NPs headed by these key terms and satisfying the regular expressions [AN]+N or [AN]*NP[AN]*N, which were shown to capture a large range of structural patterns (Justeson and Katz 1996), are considered as terms too. For example, the noun "modifier" is recognised as a key term on the basis of its frequency.

Table 1. *Examples of rules transforming a source clause to a stem*

Rule	**Source clause**	**Stem**	**Answer**
Subject-rule	The verb serves as the most central element in a clause.	Which part of speech serves as the most central element in a clause?	the verb
Which-kind-of-rule	Transitive verbs require objects.	Which kind of verbs require objects?	transitive verbs
Object-rule	A prepositional phrase at the beginning of a sentence constitutes an introductory modifier.	What does a prepositional phrase at the beginning of a sentence constitute?	an introductory modifier

The NP "introductory modifier" will be identified as a term too since it has the structure AN. Hence, the term "introductory modifier" will serve as the answer for the item to be generated from the clause "A prepositional phrase at the beginning of a sentence constitutes an introductory modifier". Although this method is very simple, it performs well in the investigated domain as only 3 items contained an answer which our post-editors dismissed as a non-term.

We also experimented with the `tf.idf` method for term extraction and noticed that important concepts in our domain like "subject", "object" and "word" were not identified as terms by this method, which gives rise to fewer items overall. As we report in our evaluation section, the time needed to post-edit an item is much less than the time needed to produce it manually. Hence, we believe that generating a larger number of items for a broader range of terms is a more appropriate strategy.

2.2 Stem generation

A clause filtering module was implemented to identify those clauses that are eligible to give rise to the stem of an item. A source clause is selected if it (i) contains at least one term, (ii) is finite and (iii) is of SVO or SV structure.[3] The module included additional genre-specific heuristics such as discounting clauses which serve as a linguistic example, contain phrases that refer to sections or tables, etc.

We utilized a few simple rules, exemplified in Table 1, to transform a source clause to the stem of an item. The subject-rule transforms an SVO clause with the term in subject position into the stem "Which H VO" where H is a hypernym of the term (retrieved from WordNet). The "which-kind-of-rule", a variant of the subject-rule, applies when the hypernym is a term too. The "object-rule" transforms an SVO clause with the term in object position into the stem "What do/does/did SV?".

The system makes use of agreement rules to ensure the grammaticality between the stem and the answer and distractors. For instance, the answer and distractors

[3] Syntactic functions and dependencies within the clause are identified by the FDG parser. We identify a clause as the text span consisting of a finite verb and the phrases that depend on it.

Table 2. *Examples of distractors accompanying generated test items*

Stem	**Answer**	**Distractors**
Which part of speech serves as the most central element in a clause?	the verb	- the noun - the adjective - the preposition
Which kind of verbs require objects?	transitive verbs	- modal verbs - phrasal verbs - active verbs

for the item in the first row of Table 2 will be in singular number. By contrast, the answer and distractors for the next item in the Table will be plural.

Dependencies on the term, as identified by the parser, are used by the system to accompany the answer and the selected distractors. For example, the answer to the item in the third row of Table 1 is not just the term "introductory modifier" but the phrase "*an* introductory modifier". We shall refer to the word "an" as the "left context" of the term.

2.3 Selection of distractors

WordNet is consulted to retrieve concepts semantically close to the answer, namely hypernyms and coordinates (i.e. concepts with the same hypernym) of the term, which can be used as distractors. If WordNet returns too many concepts, those appearing in the textbook are given preference. Examples of distractor selection using this strategy are shown in Table 2.

If the term is an NP such as "introductory modifier" for which WordNet does not return any semantically close concept, then the textbook is searched for other NPs with the same head which are used as distractors.[4] For instance, the program generates the following item using NPs headed by "modifier" as distractors:

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

Recently, we experimented with other methods to measure semantic similarity between two words.[5] They include (a) thesaurus-based methods which consider the gloss of the words in the thesaurus and the relative position of the words in a taxonomy of concepts such as WordNet and (b) distributional similarity measures that compute similarity between words in terms of the similarity of contexts of their occurrences in a corpus.

[4] In the rare case of the program being unable to extract any suitable distractor from WordNet or the text, no item is generated.

[5] These measures were used to test the distractor selection premise (see section 4.4) and have not been incorporated into the system yet.

The thesaurus-based methods included:[6] (i) the Lesk algorithm (Lesk 1986), (ii) the Leacock-Chodorow measure (Leacock and Chodorow 1998), (iii) the Jiang-Contrath measure (Jiang and Conrath 1997) and (iv) the Lin measure (Lin 1997). The distributional similarity method[7] used was the Information Radius, which according to a comparative study by Dagan *et al.* (1997) performs consistently better than other similar measures. Information Radius is a variant of Kullback-Leibler divergence measuring similarity between two words as the amount of information contained in the difference between the two corresponding co-occurrence vectors.

3 Post-editing

In this section we discuss how the test items generated by the system were post-edited. Two post-editing exercises were carried out; during the second exercise the post-editor used a user friendly environment developed to assist her in her decisions. We begin by presenting the methods used for post-editing and the utilized environment. After specifying the amount of items that could be incorporated into a test after the post-editing, we discuss how the system could be amended to account for some of the items that were rejected by the post-editor. To the best of our knowledge, this is the first work which extensively reports on post-editing automatically generated items and discusses the received feedback.

3.1 Post-editing classification

The automatically produced items had to be declared by the post-editor as either (a) "unworthy" and discarded or (b) "worthy" and either accepted without any revision or post-edited before being put into use. The post-editor was instructed to mark as unworthy items which required too much revision or did not ask about a central concept.

The items selected for further post-editing required "minor", "fair" or "major" revisions. Minor revision describes limited post-editing of the item, including operations such as insertion of an article or correction of spelling and punctuation. Fair revision refers to more extensive post-editing of the item, including re-ordering, insertion or deletion of several words and replacement of one distractor at most. Major revision involves substantial rephrasing of the stem and replacement of at least two distractors.

As an illustration, the automatically generated stem "Which kind of language units 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?" required the deletion of the string "and any theory that ... is unworthy of our attention", which was classified as a fair revision.

3.2 Post-editing environment

In the first post-editing exercise the post-editor had to revise the test items generated by the system using a simple text editor such as Notepad. To assist the

[6] The implementations in Pedersen's WordNet-Similarity package were used (Patwardhan *et al.* 2003): http://search.cpan.org/~tpederse/WordNet-Similarity/

[7] Pekar's implementation was used: http://clg.wlv.ac.uk/demos/similarity/index.html

Fig. 1. Snap-shot of the post-editing environment

post-editor in her task, a user friendly environment was developed for the second post-editing exercise. This environment is web-based and communicates with an SQL database where the revisions undertaken by the post-editor are recorded. An example of using this environment is shown in Figure 1. The test item originally produced by the system is:

(2) What do words and phrases form?
   i. the constituents of the clause
   ii. the phrases of the clause
   iii. the sentences of the clause
   iv. the optional constituents of the clause

As mentioned earlier, the system generates several distractors for each item that have to be approved by the post-editor. At the time of the first post-editing exercise the web-based environment was not available and the post-editor had to come up with an alternative from scratch, if a proposed distractor was not acceptable. By contrast, the user friendly environment utilized in the second post-editing exercise offered a list of alternative distractors that the post-editor could select from to replace the inappropriate ones. The post-editor selects a distractor from the list by ticking the box next to it. Inappropriate distractors are deselected by un-ticking the box next to them. As illustrated in Figure 1, the post-editor has selected the distractors "central constituents" and "elements" from the proposed list to replace two of the originally generated distractors (namely "phrases" and "sentences"). If none of the listed alternatives satisfies her, she can type a distractor of her choice in the "manually input" field.

The post-editing environment includes additional functionalities such as quoting the sentence which contains the source clause that gave rise to the stem of the test item, in this case "Just as words and phrases form the constituents of the clause rank, so ... the sentence". It also provides fields for revising the stem as well

as the left and the right context of the answer and the distractors. As mentioned previously, these fields consist of words which depend on the answer. For instance, the right context of the answer in example (2) originally contained the phrase "of the clause" but not the word "rank". As Figure 1 shows, the post-editor typed this word to the right context field. This word now accompanies the answer and the distractors for the item.

After the post-editor has finished revising a given item, she has to click on the "submit" button. Then, the system stores the post-edited item along with information about the level of editing (as specified by the post-editor) in the database. Each automatically generated item can be accessed by more than one post-editor at the same time and the revisions are stored separately for each editor.

3.3 Amount of unworthy items

Initially, two students reading linguistics at the University of Wolverhampton were asked to perform revisions without the help of the interface which had not been developed at the time of the first post-editing exercise. A total of 575 items were generated by the program and split between the two post-editors. They discarded 43% (that is, 247 items) as unworthy whilst the remaining 57% (328 items) were considered for further use, with or without post-editing. 20 items (3.5% of the total) were approved for direct use without any post-editing at all. The remaining 53.5% of the total (i.e. 308 items) were subjected to post-editing: 9% (52) needed minor revisions, 19.5% (111) required fair revisions and 25% (145) needed major revisions.

At this point we need to emphasise that this exercise was not designed to assess inter-editor agreement so only a subset of 90 items were judged on worthiness by both post-editors. They agreed in 67 cases (74%), which corresponds to a kappa value (Carletta 1996) of 0.44 that suggests a fair level of agreement. Investigating the agreement between post-editors in more detail using an appropriate experimental design constitutes one of our directions of future work.

A third postgraduate student edited a different set of items generated by the system using the user friendly environment. Among the 300 questions seen by this post-editor, only 30% (i.e. 90 items in total) were considered worthy. 15 of them (5% of the total) were deemed directly usable, 20 (6.7%) were subjected to minor revisions, 29 (9.6%) to fair revisions, and 26 (8.7%) to major revisions.

Notably, the number of worthy items has dropped substantially in the second post-editing exercise while, as section 4.3.3 reports, the worthy items from the second exercise appear to be of better quality than their counterparts from the first exercise. This might be due to individual differences between the third post-editor and her two colleagues who were employed in the first exercise. However, as all post-editors share similar domain knowledge and (admittedly limited) expertise in post-editing, we tentatively suggest that it is the user friendly environment that enabled the third post-editor to produce fewer, albeit better, worthy items. Despite the falling number of worthy items in the second exercise, we are satisfied that approximately 1/3 of them may still be used in the classroom. As we show in section 4.1, using the computer-aided procedure is much faster than developing

Table 3. *Examples of reasons for rejecting a test item*

Reason for unworthy	**Example**
Subordinate source clause	Unless a prepositional phrase is unusually long, ...
Uninformative source clause	The verb is recognizable by several functional properties.
Anaphoric phrase	Adverbial complements occur only in those positions.
Stem contains answer	A verb phrase is normally headed by a verb.
Non-generic source clause	The context of situation is different for each sentence.

items manually, even though a large number of automatically generated items need to be rejected. This holds for all post-editors and both post-editing exercises.

3.4 Error analysis and future system development

In order to further develop the system on the basis of the feedback we received from the post-editors, we analysed 140 randomly selected unworthy items in an effort to identify the reason that they might have been rejected. The most evident reasons for unworthiness of the investigated items are shown in Table 3.

In 23% of the analysed cases, the system produces an item using a subordinate clause as its source. This gives rise to a stem such as "Which kind of phrase is unusually long?" from the source clause in the first row of Table 3. Uninformative source clauses contain trivial or unimportant information and were found to account for 16% of unworthy items, whilst source clauses which contain anaphoric elements such as "those positions" stand for another 9%. The fourth row of Table 3 shows a source clause which contains the string "a verb phrase" which hints at the answer of the corresponding item (i.e. "a verb"). This was another frequent reason for unworthiness (8%). Last but not least, the statement in the next row of the Table only applies to the specific example sentences discussed within the section that contains the cited source clause (rather than each and every sentence in English). Non-generic source clauses like that represent another 10% of unworthiness.

Just 5% of the investigated items were found to be irreparable due to ungrammaticality caused by the rules that turn the source clause into a stem. Moreover, the more complicated the structure of the source sentence, the more difficulties the parser had in producing a correct analysis. However, only 4% of the items were so ungrammatical due to a severe parsing failure that they were deemed unworthy. These cases had mainly to do with source sentences containing relative or cleft clauses and fronted or dislocated material. Similarly to the mild problems caused by the transformational rules, most parsing failures were accounted for by the post-editors' actions which added or removed material during the revision stage.

Several unworthy items (17%) were found to suffer from more than one problem, whereas we have not been able to specify why another 8% were rejected using the categories presented here. Even so, from this analysis we concluded that the unworthiness of an item has largely to do with the inappropriateness of its source clause.

Even though excluding subordinate source clauses from giving rise to a stand-alone item can be implemented in a rather straightforward way within our existing clause filtering module, accounting for the other reasons exemplified in Table 3 might require more sophisticated techniques. Expanding the sentence filtering module to provide for these cases is our main direction of future work.

Although we did not conduct an exhaustive analysis of the actions undertaken by the post-editors, most of them involved the following operations: (a) removing discourse words such as "however" and "therefore" from the stem, (b) turning words from uppercase letters into lowercase and vice-versa, (c) correcting the punctuation in the stem, (d) closing parentheses where appropriate and (e) correcting the number agreement between the stem and the answer or the distractors. We are confident that we will be able to account for most of these cases within the existing modules of the system as it is further developed.

Despite the fact that the scheme which supports our error analysis is preliminary, it appeared to supply us with useful hints as to why an item is considered to be unworthy and how the system could be amended to prevent this from happening. In our current work, we are extending this scheme to cover more cases of unworthy items as well as the most common post-editing actions. This will include conjectures on what makes a distractor inappropriate.

4 Evaluation

As reported in section 3.3, the system produces a fair amount of items which can be later post-edited and incorporated into a test. The proposed methodology was evaluated in two ways. Firstly, we investigated the efficiency of the procedure by measuring the average time needed to produce a felicitous item with the help of the program as opposed to the average time needed to produce it manually. Secondly, we assessed the quality of the semi-automatically generated items, and compared it with the quality of manually produced items using standard test theory measures. In the next two sections we report on our evaluation efforts in more detail.

4.1 Efficiency of computer-aided test item generation

For each post-editing exercise, we calculated the overall time it took for the post-editor to perform the task, including the rejection of unworthy items. In the first exercise, this time was 540 minutes for both post-editors which, divided by the total of 328 worthy items (as these items represent the end-product of the whole procedure), yielded an average of 1 minute and 36 seconds per worthy item to be constructed.

After the first exercise, the two post-editors involved were asked to manually produce 65 items from another chapter of the Linguistics textbook. A different chapter than the ones used as the inputs to the system was chosen to ensure that the post-editors were not familiar with its content or biased by the post-editing task they had recently undertaken. However, being part of the same textbook, the writing style and the difficulty of this chapter was similar to the ones used as the input to the system. This task took the post-editors 450 minutes resulting in 65 manually constructed items and an average of 6 minutes and 55 seconds per

item. Clearly, the average time the post-editors spent on producing items using the computer-aided procedure during the first exercise compares favourably with the average time they spent when producing items manually.

In the second post-editing exercise, the third post-editor produced 90 worthy items within 150 minutes with the help of the user friendly environment. This averages to 1 minute and 40 seconds per worthy item, which is very similar to the time recorded for the post-editors in the first exercise.

The same person was asked to manually produce 40 items from a chapter that was used as the input to the system. This was done more than a year after she performed the post-editing task so any effect of familiarity with the material was practically eliminated. It took the third post-editor 240 minutes to manually produce these items which results in an average of 6 minutes per item. This is comparable to the time recorded for her colleagues in the first exercise.

Therefore, although using the user friendly environment did not result in faster post-editing times for the third post-editor when compared to her two colleagues in the first exercise, her average time per worthy item continues to compare favourably with the time it took her to produce items manually. Hence, computer-assisted test item generation is shown to be much faster than manual production of items in both post-editing exercises.

4.2 In-class experiments

Controlled sets of the post-edited items were used to test students and obtain evaluation data related to their quality. Only items approved by a linguistics lecturer were used to make sure that the items addressed material taught to the students. Two experiments were conducted: In the first experiment, we employed 24 test items constructed in the first post-editing exercise (without the user friendly environment) as well as another 12 manually produced ones. In the second experiment we made use of 18 items produced in the second exercise (with the help of the environment) as well as the same 12 manually produced items from the first experiment.

The selected items were delivered via Questionmark's Perception testing software. Perception has a web-based interface which generates the position of the correct answer randomly for each item and makes the test accessible to the students no matter where they are located. The first experiment was conducted in class and the participants were supervised. In the second experiment, the students accessed the interface from their own browser and conducted the test without supervision. In both cases the software would accept answers only from the students who completed the test within 30 minutes. We acquired data from 30 undergraduate students in linguistics for the first experiment and 78 for the second.

4.3 Item analysis

Item analysis is a popular procedure in classical test theory which provides information as to how well each item has functioned in the multiple-choice test. The standard method for item analysis consists of the following variables (Gronlund 1982):

(i) the difficulty of the item, (ii) its discriminating power and (iii) the usefulness[8] of each distractor. Item analysis tells us if a specific item was too easy or too hard, how well it discriminated between high and low scorers in the test and whether the distractors functioned as intended.

In order to conduct the analysis, we used the simplified procedure described in Gronlund (1982). First, we arranged the test papers in order from the highest score to the lowest score. Then we selected one third of the papers with the highest scores and called this the upper group (10 papers in the first experiment, 26 in the second). We also selected the same number of papers with the lowest scores and called this the lower group. 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 and computed the following scores for each item:

Item Difficulty (ID) is estimated by establishing the ratio of students from both groups who answered the item correctly (ID = C/T, where C is the students who answered the item correctly and T is the total number of students).

Discriminating Power (DP) is estimated by comparing the number of students in each group who answered the item correctly (based on the formula DP= (C_U − C_L) ÷ (T/2), where C_U is the number of students in the upper group who answered the item correctly, C_L is the number of the students in the lower group who did so and T stands for the total number of students). It is desirable that the discrimination power is positive which means that the item differentiates between students in the same way as the total test.[9]

Distractor usefulness (DU) is estimated by comparing the number of students in each group who selected each distractor. A good distractor should attract more students from the lower group than the upper group. Distractors are classed as poor if they attract more students from the upper group than from the lower group.

4.3.1 Results of the first in-class experiment

As Table 4 shows, the average ID of the items generated with the help of the computer in the first in-class experiment is 0.75. The average ID of the items which were produced manually is 0.59, which is closer to the recommended score of 0.50. An independent samples t-test with ID as the dependent variable and "computer-aided" versus "manually produced" as the contributing factors showed that this difference was significant beyond a level of confidence of 0.05 (t=1.73, df=34, p<0.05). Addi-

[8] Originally called "effectiveness". We chose to term this variable "usefulness" to distinguish it from time-effectiveness of the semi-automatic procedure as opposed to the manual construction of tests (see section 4.1).

[9] Zero DP is obtained when an equal number of students in each group answer the item correctly. Negative DP is obtained when more students in the lower group than the upper group answer it correctly. Items with negative DP should be discarded. Maximum positive DP (1.0) is obtained when all students in the upper group answer correctly while no one in the lower group does so. An item with maximum DP will have an ID of 0.5; therefore, test developers are advised to construct items at an ID of 0.5.

Table 4. *Item analysis for the first in-class experiment*

Item difficulty:	average	stand. dev.	item #
computer-aided	0.75	0.24	24
manually produced	0.59	0.30	12
Discriminating power:	average	stand. dev.	item #
computer-aided	0.40	0.25	24
manually produced	0.25	0.20	12
Distractor usefulness:	average	stand. dev.	distractor #
computer-aided	1.92	1.40	65
manually produced	1.18	1.80	33

tionally, there were 3 too easy items among the 24 computer-aided ones (12.5% of the total), but only 1 between the 12 manually produced (8.3%).[10]

The average DP of the items generated with the help of the computer in the first experiment was 0.40, with the manually produced items scoring just 0.25. This difference is also significant ($t=1.81$, $df=34$, $p<0.05$), showing that the computer-aided items perform better than the manually produced ones on this variable. There was only 1 item that discriminated negatively among the computer-aided items (4.2%), but 2 among the manually produced ones (16.7%).

The evaluation of the distractors estimated the average DU to be 1.92 for the computer-aided items and 1.18 for the manually produced ones, another significant difference ($t=2.24$, $df=96$, $p<0.05$). There were 6 poor distractors in the computer-aided items (9.2%) which compares favourably to the 10 poor distractors in the manually produced ones (30%).[11] Finally, 3 distractors among the computer-aided items (4.6%) were deemed not useful because they were not selected by any student. The manually produced items included 2 not useful distractors (6.1%).

Overall, manually produced items were found to do significantly better than the ones generated with the help of the computer only with respect to their ID in the first experiment. Crucially, the material generated by the computer-aided procedure performed significantly better than the manually produced items as far as their DP and DU are concerned.

4.3.2 Results of the second in-class experiment

The average ID of the 18 items which were post-edited with the help of the user-friendly environment and utilized in the second in-class experiment was 0.58, which is much closer to the recommended value and not significantly different from the ID score of the manually produced ones ($t=0.36$, $df=28$, $p>0.05$). There were 2 too easy items among the manually produced ones (16.7%) but none otherwise.

The average DP of the computer-aided items in the second experiment was 0.36, with the manually produced ones scoring just 0.26. This difference approaches sig-

[10] We consider an item to be "too easy" if its ID is higher than 0.85 and "too difficult" if it is lower than 0.15.

[11] Note that 7 computer-aided and 3 manually developed items had only 2 distractors assigned and this is why the total number of distractors is 65 and 33 respectively.

Table 5. *Item analysis for the second in-class experiment*

Item difficulty:	average	stand. dev.	item #
computer-aided	0.58	0.16	18
manually produced	0.56	0.30	12
Discriminating power:	average	stand. dev.	item #
computer-aided	0.36	0.17	18
manually produced	0.26	0.16	12
Distractor usefulness:	average	stand. dev.	distractor #
computer-aided	2.94	2.40	54
manually produced	0.90	1.50	33

nificance at a confidence level of 0.05 (t=1.62 df=28, 0.10>p>0.05). Neither the items produced with the help of the computer nor the manually constructed ones contained any items with negative DP.

The average DU was 2.94 for the computer-aided items and only 0.90 for the manually produced ones, a very significant difference (t=4.38, df=85, p<0.001). There were 3 poor distractors in the computer-aided items (5.5%) which compares favourably to the 5 poor distractors in the manually produced ones (15%). Finally, only 1 distractor among the computer-aided items (1.8%) was deemed not useful while there were 8 not useful distractors in the manually produced items (24%).

In summary, the items in the second experiment which were produced with the help of the post-editing environment did: (i) not differ significantly from the manually prepared ones in their ID, (ii) perform slightly (yet not clearly significantly) better than the manually prepared ones in their DP and (iii) were overwhelmingly better than the manually constructed ones with respect to their DU.

4.3.3 Comparisons between experiments

Although the same set of manually produced items were viewed by two different groups of students, the average scores for ID, DP and DU for these items did not differ significantly between the two experiments (ID: t=0.24, df=22, p>0.05; DP: t=0.13, df=22, p>0.05; DU: t=0.68, df=64, p>0.05). However, the items produced with the assistance of the system did differ in the two experiments especially with respect to their ID and DU, for which the tests identify very significant differences as indicated by the bold font in the reported results (ID: t=2.44, df=40, **p<0.01**; DP: t=0.58, df=40, p>0.05; DU: t=2.88, df=117, **p<0.005**).

Based on the similar performance of the students on the manually produced items, we conclude that the improved scores in ID and DU for the items generated with the help of the computer in the second experiment are not caused by the different set of subjects used in each experiment. Following the same reasoning as in section 3.3, we conjecture that this result may be explained by the addition of the user friendly environment in the second post-editing exercise.

4.4 Evaluation of the distraction selection premise

In section 2, we put forward the premise that good distractors are those that are as semantically close as possible to the correct answer. In this section we partly eval-

Table 6. *Example of usefulness and different similarity measures*

Distractor	**Answer**	**Usefulness**	**jcn**	**lch**	**lesk**	**dis-bnc**	**dis-dom**
modifier	verb	0	0.6	2.36	188	0.02	0.30
relative	reflexive	5	0	0	8	0.03	0.48
sound	grammar	3	0	0.75	44	0.03	0.50
case	tense	-1	0.26	2.36	145	0.10	0.28

uate this hypothesis by assessing the correlation between the degree of usefulness of the distractors and their semantic distance to the correct answer. As already explained, we use a simple measure for usefulness expressed by the difference in the number of students from the lower group and students from the upper group who selected a specific distractor. Negative figures indicate that distractors are poor, whereas positive figures point to good distractors.

The function *coordinate terms* in WordNet was used to compute semantic similarity (with a preference towards the terms which appear in the materials themselves for single word terms), while same head noun-phrases are also used as distractors for multi-word terms. In this experiment we employed other semantic closeness techniques based on WordNet- or corpus-based distributional similarity measures to assess the correlation. The WordNet-based semantic similarity scores included jcn, lin, lch and lesk (see section 2.3), whereas Pekar's program was deployed to compute distributional similarity, against the BNC and the domain-specific corpus used to generate test items (see Table 6 for some examples of those scores).

As Pekar's program operates only on single-word units, for this experiment, only two types of pair of distractor and correct answer are used: 1) single-word distractors and correct answers; and 2) single-word modifiers of the distractors and correct answers (for example, if the correct answer is "reflexive pronoun" and the distractor is "relative pronoun", the pair "reflexive" and "relative" are chosen for similarity calculation. Because of this (single-word unit) constraint, only 26 (out of 65) pairs of distractor-correct answer are used.

In order to establish the correlation between usefulness and semantic similarities, the cosine distance and a linear regression model were used. The results obtained (Table 7) confirm the premise that the best distractors are the ones that are semantically closest to the correct answer. As we can see from the Table, while Lesk's and lch's semantic similarity measures seem to be the most suitable among the WordNet based ones, the domain-specific corpus-based distributional similarity emerges as the most correlated to the usefulness of the distractors. This suggests that the distributional approach should be seriously considered as an alternative for choosing plausible distractors.

4.5 Discussion

The evaluation results clearly show that the construction of multiple-choice test items with the help of our program is much more efficient than purely manual production. We believe that this is the main advantage of the proposed methodology. As an illustration, the development of a dataset of considerable size consisting of

Table 7. *Cosine distance between usefulness and different semantic similarity scores, calculated against 26 pairs of correct answer and generated distractors*

Cosine distance	**jcn**	**lin**	**lch**	**lesk**	**dis-bnc**	**dis-dom**
	0.35	0.40	0.56	0.62	0.45	0.72
Linear regression	**jcn**	**lin**	**lch**	**lesk**	**dis-bnc**	**dis-dom**
R-squared	0.028	0.009	0.004	0.122	0.027	0.001
p-value	0.35	0.60	0.71	0.05	0.37	0.084

1000 items would require approximately 30 hours of human work when using the program, and 115 hours if done manually. This has direct financial implications as the time and cost in developing items would be dramatically cut. Although the post-editing environment was not actually found to make post-editing faster, its functionalities are intended to facilitate this process.

At the same time, the item analysis shows that the quality of test items produced semi-automatically is not compromised in exchange for time and labour savings. This was particularly true for the items which were post-edited with the help of the user friendly environment which strengthens our confidence in its usefulness. As a matter of fact, the post-edited items often scored better than those produced manually. Whereas manually constructed items return better values for ID than the items produced with the help of our system, the latter scored better on DP.

The analysis of the distractors showed that the values for DU for the distractors generated with our computer-aided procedure were significantly better than for the manually selected ones. Crucially, when the distractors were selected via the user friendly interface the effect was even stronger. This result is particularly important given that the selection of distractors is the most challenging task in the generation of test items, which can be greatly eased now using our methods. Finally, it is worth noting that the evaluation of the distraction selection premise confirmed that the best distractors are those which are semantically closest to the correct answer.

5 Ongoing and future work

It should be noted that the post-editors we employed were not professional test developers. We are currently investigating the performance of the system in the medical domain in collaboration with such professionals.

The error analysis discussed in section 3.4 encourages further amendments to the system especially as far as the clause filtering module is concerned. Forthcoming work will also include an extensive corpus-based study for the genres covered whose findings will help us enhance our existing classification scheme.

Moreover, we intend to experiment with more advanced term extraction techniques and other more elaborate models for measuring semantic similarity between concepts. Evaluation as to the extent to which the questions cover the course material is envisaged too. Finally, even though the agreement between post-editors appears to be a complex issue, we would like to investigate it in more depth.

Acknowledgements

We are grateful to Claire Wevil, Georgina Miller and Laura Hasler for their post-editing work. Special thanks to Victor Pekar for his input on the section about distractor selection and to two anonymous reviewers as well as Richard Evans for their comments.

References

Brown J., Firshkoff G. and Eskenazi M. (2005). Automatic question generation for vocabulary assessment. *Proceedings of HLT/EMNLP-2005*, 819-826. Vancuver, Canada.

Carletta J. (1996). Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics* 22 (2): 249-254.

Dagan I., Lee L., and Pereira F. (1997). Similarity-based methods for word sense disambiguation. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 56-63. Madrid. Spain.

Fairon C. (1999). A web-based system for automatic language skill assessment: EVALING. *Proceedings of the ACL Workshop on Computer Mediated Language Assessment and Evaluation in Natural Language Processing*, 62-67. Maryland, US.

Gronlund N. (1982). *Constructing Achievement Tests*. Prentice-Hall Inc, NY.

Hoshino A. and Nakagawa H. (2005). Real-time multiple choice question generation for language testing: a preliminary study. *Proceedings of the Second Workshop on Building Educational Applications using Natural Language Processing*, 17-20. Ann Arbor, US.

Jiang J. and Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *Proceedings of the International Conference on Research in Computational Linguistics*, 19-33. Taiwan.

Justeson J. and Katz S. (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 available from http://www.papyr.com/hypertextbooks/

Leacock C. and Chodorow M. (1998). Combining local context and WordNet similarity for word sense identification. In Fellbaum, C. (Ed.) *WordNet: An Electronic Database*, pp. 265-283. MIT Press.

Lesk M. (1986). Automatic sense disambiguation: How to tell a pine cone from an ice cream cone. *Proceedings of the SIGDOC'86 Conference*, 24-26. Toronto, Canada.

Lin D. (1997). Using syntactic dependency as a local context to resolve word sense ambiguity. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, 4-71. Madrid, Spain.

Liu C., Wang C., Gao Z. and Huang S. (2005). Applications of lexical information for algorithmically composing multiple-choice cloze items. *Proceedings of the Second Workshop on Building Educational Applications using Natural Language Processing*, 1-8. Ann Arbor, US.

Mitkov R. and Ha L.A. (2003). Computer-aided generation of multiple-choice tests. *Proceedings of the First Workshop on Building Educational Applications using Natural Language Processing*, 17-22. Edmonton, Canada.

Patwardhan S., Banerjee S. and Pedersen T. (2003). Using measures of semantic relatedness for word sense disambiguation. *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics*, 241-257. Mexico City.

Sumita E., Sugaya F. and Yamamoto S. (2005). Measuring non-native speakers' proficiency of English using a test with automatically-generated fill-in-the-blank questions. *Proceedings of the Second Workshop on Building Educational Applications using Natural Language Processing*, 61-68. Ann Arbor, US.

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