GeS 2007 German e-Science

Available online at http://www.ges2007.de

Sentence Completion Tests in a Virtual Laboratory

M. Hess and C. Mahlow

Institute of Computational Linguistics, University of Zurich, Binzmühlestrasse 14,
CH-8046 Zürich, Switzerland
email: {hess, mahlow}@cl.uzh.ch
phone: +41 44 635 43 77, *fax:* +41 44 635 68 09

Abstract

This Paper describes a type of on-line test, the Sentence Completion Test (SCT), that tries to fill the gap between rigid Multiple Choice tests and unreliable automatic essay grading approaches. We give a short overview of the main concepts, the implementation and show examplary uses and applications. SCTs are used as one component in a fully operational virtual laboratory of Computational Linguistics in use at the University of Zurich.

1 Introduction

One of the least satisfactory aspects of contemporary e-learning systems are the limitations of on-line tests with automatic assessment. Basically, there are MC tests (and their derivatives, like true/false and fill-the-gap tests) on the one hand, and systems trying to rate free form text (automatic essay grading systems like *e-rater* ([7]) on the other. The first type of tests are easy to create for authors and easy to use for students but they can basically test only the presence or absence in the students' mind of small and relatively isolated fragments of knowledge. This is fine for testing factual knowledge or comprehension but more general and abstract knowledge or even application, analysis and synthesis (according to Bloom's taxonomy of educational objectives [1]) is hard to assess that way. For this you would prefer the second type of test, automatic essay rating. Given the limits of existing Natural Language Processing technologies this approach is, however, limited to an extremely superficial analysis of the texts submitted by students, and the resulting assessment is thus highly problematic despite initially high correlations between human and machine assessments – once users know how the system works cheating becomes very easy (see [3]).

In this paper we show a solution to overcome this dilemma. We developed a type of on-line test, the Sentence Completion Test (SCT), that tries to fill the gap between rigid MC tests and unreliable automatic essay grading approaches. Our (fully implemented and operational) system (see [2]) allows students to interactively compose answers to questions through a sequence of menu choices each of which presents several possible continuations of the answer. Each continuation is thus a fragment of a possible answer. That way users compose, in

a step-wise manner, potentially very complex answers to questions about potentially very high-level concepts of the domain. Once the answer is complete the system generates a comment which can, again, be arbitrarily complex and detailed. Additionally the system may rate the answer in terms of marks.

We present the main concepts of Sentence Completion Tests, describe the existing implementation, and mention some instances of such tests as used in the domain of Computational Linguistics as well as possible applications in completely different domains.

2 Main concepts

The main idea behind this novel type of test is the observation that answers to questions (and their components) fall into a rather small number of well-defined types. There may be considerable linguistic variability in the concrete phrasing of answers but when we abstract away from these surface phenomena we end up with a relatively simple semantic structure. This structure allows us, in a second step, to generate a number of comments for each concrete answer (and their components) by defining a small number of logical conditions on certain properties of this structure. We end up with a system that can accept a large number of answers and generate a large number of comments, but in a strictly defined formal framework and therefore with quite limited effort on the part of test authors ([5]).

2.1 Guided Input of Answers

Each sentence completion test defines a (potentially infinite) number of answers to a given question by way of a finite state automaton (FSA), possibly containing cycles. The *states* of the automaton represent individual propositions about crucial concepts in the domain of the question. The *transitions* between states represent the linguistic manifestations of the proposition represented by the target state. Often there will be several non-trivially equivalent ways of describing a concept. In such cases there will be several transitions *to* the state representing this concept, one for each linguistic variant. In most cases there will be several ways to continue an answer once a given state has been reached. In such cases there will be several transitions *from* this state.

In the simplest case there will be one single path through the FSA defining the one and only *completely true* answer, possibly with some minor linguistic variations (such as synonyms of terms), and several paths defining a number of *completely wrong* statements. More often there will be several completely true answers, each with its own linguistic variations. In more interesting cases paths will have to be defined for answers that are *correct in principle, but too general*, and conversely for those that are *nearly correct but too specific*. By way of example, the question "What is a computer?" can be answered by a statement "Any *device* that ... processes information ...". However, phrased that way, this statement would also cover natural systems such as brains, which is

too general to be unconditionally correct. What we want is a statement like "Any *technical system* that ... processes information ...". On the other hand, the answer "Any *electronic* system that ... processes information ..." is too specific to be unconditionally correct (computers need not be electronic, they may well be mechanical). It proved very important to distinguish between these cases of answers that are neither completely true nor completely wrong. See figure 1 for the FSA representing this example. Note that this FSA is not minimal, by design, in order to distinguish syntactically equivalent but semantically different propositions. This is why states 1, 2, and 3 are distinguished.

Often we will need sub-paths defining *redundant* and/or *inappropriate* information. They do not make an answer wrong but are not part of the correct statement. Thus it is redundant to say that a computer is an information processing system that processes data, while it is inappropriate to say who invented the computer when the question asked for a definition of the term "computer".

In some cases even cycles in the FSA are useful, in particular to allow the enumeration of items (conjoined with "and" or "or"), something that is often required in questions about the structure of systems ("What are the components of a computer and the connections between them?").

2.2 Automatic Creation of Comments

Once an answer has been composed by the user, a comment is generated. The comment does not merely rate the answer ("correct", "wrong") but explains in detail in what way the answer was correct or wrong, what was missing or redundant etc.

The point now is that the accuracy (correctness, conciseness, and appropriateness) of each answer can be described by the test author through first order logical statements ranging over the individual states of the FSA traversed when composing the answer. Once an answer has been completed, the various comment fragments are collected and concatenated in a sequence that respects all logical constraints. That way a usually rather small number of conditions on states will define a large number of comments produced by the system even if the answers can become long and complex.

- In the simplest case (completely correct answer), a comment condition merely tests whether all the required true states have been traversed. If so, the comment may be a simple "Correct". In some cases one *single* crucial proposition is sufficient for a correct answer, and then the corresponding state in the FSA is itself assigned the comment "Correct". In most cases, however, *several* propositions must be contained in the answer for it to be correct, and then the comment "Correct" is elicited only if *all* the corresponding states have been traversed. This can be enforced by a simple logical *conjunction* over the corresponding true states.
- If one or more states have been traversed that represent explicitly *wrong* concepts (wrong answer) the comment ought to be more informative (not just "Wrong") and explain in what way(s) the answer is wrong (spelling out the correct answer). This is encoded by a simple logical *disjunction*

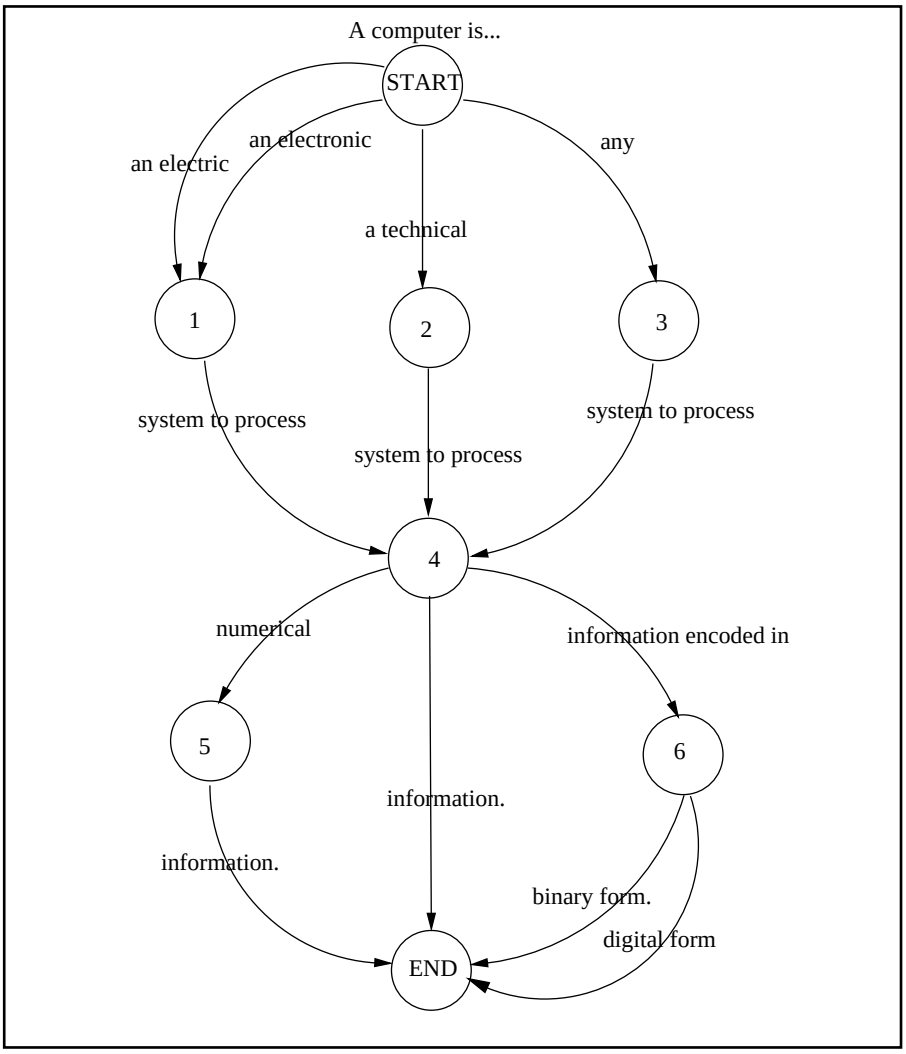

Figure 1: A very simple answer FSA

over the corresponding false states to first output the general comment "Wrong" (a single serious error makes the answer wrong) plus additional conditions for each false state to output specific explanations for each of the errors.

- If *only* true states have been traversed *but not all* those that are required (partially correct answers), we must again concatenate comment fragments. First, we will want to output a comment like "Correct to a point." and then continue with something like "However, you forgot to mention ...". This latter fragment will be linked with a negative logical

condition for each missed correct answer component. It is didactically important to mention the missed parts in order of decreasing importance. This can be encoded as additional logical conditions for each fragment.
- Answers that are correct but contain redundant or irrelevant material (redundantly correct answer) can also be covered (by a logical condition testing for the existence of states that are neither required nor explicitly wrong). Again, the most important comments will have to be output first.
- Even those (none too rare) cases where a user has run in cycles but finally came up with the right answer (cyclically correct answer) can be covered by logical conditions on comment fragments.
- And, finally, if parts of an answer are in mutual contradiction (inconsistent answers) this, too, can be determined by a logical condition, and can be commented upon accordingly.

It is important that SCTs can determine *non-local properties* of answers. Thus a SCT can detect contradictions between different parts of an answer, or a wrong order in the description of processing steps, or needless repetitions, all of which may occur in parts of an answer that are arbitrarily far removed from each other.

2.3 Interactions between User and System

The structure of the sentences that can be input is completely determined by the FSA, and the kinds of comments generated by the system are equally determined by the logical conditions on comment fragments. What is not defined that way but has a direct bearing on the usability of the system, are the types of interaction allowed between user and system. By way of example, it proved useful to give users a look-ahead of one step. They can, in other words, see what path will open if they opt for a certain continuation. Without this feature they tend to feel lost in the space of possibilities. This is a typical case of external condition, as implemented in the interface.

It proved equally useful that users can backtrack a step when they realise they are completely off track. Naturally, this makes it possible to mindlessly trying out all possible answers until a satisfactory comment has been generated. Thus, if a user found the right answer by wildly trying around (random correct answer) they can be reprimanded suitably (backtracking is recorded in the path, and a suitable condition will catch these cases).

Finally, it is very useful to send users back to a specific state of the the FSA if their answer was only partially correct. That way they can acquire the missing piece of information without having to run through the entire FSA again. If their answer betrays a more fundamental lack of understanding they can be sent to the very beginning of the test or, in more serious cases, to the exact part of a text they should read in order to fill the gaps in their knowledge. Even specific tests or other interactive components can be triggered by certain types of answers. That way it is possible to create completely customised learning paths for user groups with heterogeneous backgrounds.

3 Implementation

The viability of this concept hinges on its implementation and, in particular, on the user-friendliness of the interface. After some experimentation we found it useful to have the system interact with the user through four windows.

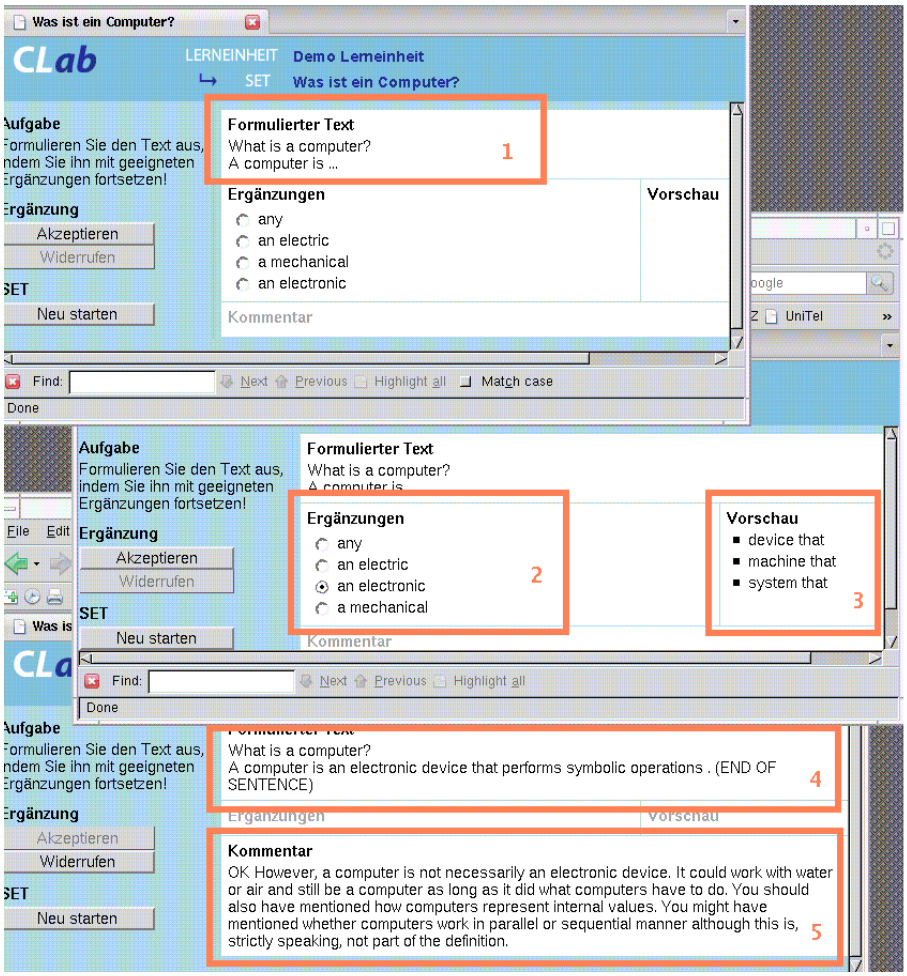

Figure 2: An SCT in action

The interaction is started by the presentation of the question and the first few words of the answer in the *protocol window* ("Formulierter Text") at the top (number 1 in figure 2). At the same time, in the *main menu window* just below the protocol window ("Ergänzungen"), all possible continuations of the answer are presented, in randomised order, as a menu. As soon as the user chooses one answer fragment from the menu (number 2), the possible continuations, one step ahead, are shown in the *preview window* ("Vorschau") to the right of the main menu window (number 3). If the user decides, in view of the possible continations shown, that his choice of answer fragment was wrong he may click on a different menu item, and the appropriate new continuations are shown. Once the user is satisfied that his choice is fine, he confirms it (by clicking on the button "Akzeptieren") and this fragment of the answer is added to the protocol window, whereupon the next choice opens up in the main menu window. That way the answer, interactively composed by the user through his menu choices, steadily grows in the protocol window, until an end state of the FSA is reached and the answer is complete (number 4). At this point the complete comment is output by the system to the *comment window* ("Kommentar") at the bottom (number 5), and the test is over.

The test system itself is server based (programmed largely in Prolog). The user interface is purely browser based and, hence, platform independent. The authoring interface is Java based, with packages for Solaris and Windows. Work is underway to integrate the entire test system in an existing Learning Management System (OLAT). OLAT would allow us to log individual users' activities, such as who worked through which SCT and how often, and what were the results. Such data could be used to better guide the user through the learning space. Depending on how they fared with respect to a given SCT, they would be steered either towards more difficult or towards easier learning units, helping them to acquire supplementary information or fill the gaps in their knowledge, respectively. And if there were several SCT covering the same topic the user could be shown a different SCT each time they went through the learning unit.

4 Uses and applications

The richness of the comments that can be generated by SCLs with limited effort on the author's part makes these tests particularly useful for self-assessment and training. As users get elaborate feedback for all aspects of their answers they can gradually explore the entire space of possibilities for a given question. This proved particularly useful for questions where there is no absolute consensus in the community about the correct answer to the question (and this is the case puzzingly often, even in a highly technical domain such as computer science). In such cases students ought to be able to assess the merits of the various competing "schools of thought", something that can be trained through the use of SCTs.

SCTs are used as one component in a fully operational virtual laboratory of Computational Linguistics in use at the University of Zurich, the *CLab*[1]. Close to 100 SCTs have been designed so far. Other components of the laboratory are demos and interactive experiments ([4]), an on-line glossary, and extensive multi-layered interactive texts. The laboratory as a whole is used in the newly established BA courses in Computational Linguistics and Language Technology at the University of Zurich.

4.1 Existing applications

In the courses and self-studying units for Computational Linguistics we use SCTs for:

- definitional questions about objects ("What is a finite state automaton?", "What is a parser?")
- questions about methods and processes ("How does a compiler work?")
- questions requiring descriptions of specific procedures ("What are the processing steps of a transfer based Machine Translation system?")
- questions requiring comparisons between concepts ("How does a parser differ from an acceptor?")

We also "abuse" SCTs to function as a formula editor with elaborate feedback facilities. Students can, for instance, be asked to write regular expressions for a specific task[2]. They get, at each step, the complete set of symbols available (constants, variables, parentheses, Kleene star, etc.) and have to combine them, step by step, creating an expression in the process. If their choice of a symbol is completely off track (such as an expression beginning with a closing parenthesis) they are warned right away. Otherwise the structure of the complete expression is commented upon at the end. If the expression is not correct, users are given an explanation and are sent back to the beginning. Otherwise they are sent to a subsequent SCT, with a more demanding task. That way, by chaining SCTs, we teach them to write increasingly complex expressions, under close guidance of the system. According to [1] students are guided to achieve learning objectives at the level of *synthesis* and even *evaluation*. This turned out to be a very promising use of SCTs.

4.2 Future Applications

While SCTs have been used so far for one domain only (Computational Linguistics), any type of question in any field that would, ideally, require a free form answer are candidates for a Sentence Completion Test. The types of questions are equivalent to the ones mentioned in 4.1:

- questions about natural or artifical objects – "What is a ligand?" (requiring *knowledge* and *comprehension*[3])

[1] `http://www.cl.uzh.ch/clab`

[2] e.g. to create complex rules for chunk-parsing:
`http://arvo.ifi.uzh.ch:9090/clab/chunking/set_chunk_01/index.jsp`

[3] all according to [1]

- questions about methods and processes – "How does the endogenous circadian clock activate transcription of which genes?" (requiring *knowledge* and *comprehension*)
- questions requiring descriptions of specific procedures – "Which strategies exist to solve diplomatic problems between two countries concerning an airport near the border?" (requiring *application, analysis* and *synthesis*)
- questions requiring comparisons between concepts – "How does nuclear fission differ from nuclear fusion?", "Why doesn't the Sun form an iron core?" (requiring *application, analysis* and *synthesis*)

A concrete promising application might be the interpretation of medical images (scans, X-rays etc.). Describing what one sees on a scan and giving it the most likely interpretation is very similar to writing an essay on a given topic. Here, too, there are various ways to go about the task and the order in which various features are described may be immaterial, while it is crucial that every single important feature is described and interpreted correctly. Again, the description may contain irrelevant or even distracting detail while covering all the important ones. Or it may, of course, be blatantly wrong in certain vital respects. An SCT might be very useful in checking the user's ability to describe and correctly interpret all the, and only the, important features in the image.

5 Conclusion

A careful combination of simple formal devices (Finite State Automata and First Order Logic) makes it possible to create tests that are far more powerful than standard Multiple Choice tests yet can be assessed in a fully automatic manner. They are suitable for self-assessment and training as well as for exams. They can be used in basically any domain where textual essays could be used for testing purposes.

Acknowledgements

Our thanks go to Simon Clematide, our superb project manager, to our teaching assistants (in chronological sequence) Sonja Brodersen, David Lee, Sandra Roth I, Sandra Roth II, Luzius Thöny, Michael Amsler and Christos Bräunle, who developed and implemented the whole system, and to Esther Kaufmann, who created most of the existing SCTs.

References

1. B. Bloom, M. Englehart, E, Furst, W. Hill, and D. Krathwohl. "Taxonomy of educational objectives: The classification of educational goals. Handbook I: Cognitive domain." New York, Toronto: Longmans, Green. 1956
2. Sonja Brodersen, David Lee. Dynamisches Multiple-Choice mit Satz-Ergänzungstests. Dokumentation zum gesamten Satztestprojekt. Unpublished, December 2004.

3. Jill Burstein. The e-rater scoring engine: Automated essay scoring with natural language processing. In M. D. Shermis and J. Burstein, editors, “Automated essay scoring: A cross-disciplinary perspective.” Lawrence Erlbaum Associates, Inc. Hillsdale, NJ. 2003
4. Kai-Uwe Carstensen, Michael Hess. Problem-based web-based teaching in a computational linguistics curriculum. “www.linguistik-online.de”, 17(5/2003).
5. Cerstin Mahlow, Michael Hess. Sentence Completion Tests for Training and Assessment in a Computational Linguistics Curriculum. COLING-2004 Workshop on eLearning for Computational Linguistics and Computational Linguistics for eLearning, Geneva 2004. 301-319
6. Cerstin Mahlow, Michael Hess. Satzergänzungstests im web-basierten virtuellen Laboratorium der Computerlinguistik. Poster at 11. Jahrestagung der Gesellschaft für Medien in der Wissenschaft 2006, 19. bis 22.9.2006, Zürich
7. Donald E. Powers, Jill Burstein, Martin Chodorow, Mary E. Fowles, and Karen Kukich. Stumping E-Rater: Challenging the Validity of Automated Essay Scoring. “GRE Research, GRE Board Professional Report No. 98-08bP, ETS Research Report 01-03”. 2001