GLIMPSE - A Statistical Adventure

David E. Wolstenholme

Imperial College of Science and Technology,
Department of Computing,
180 Queen's Gate, London, SW7 2BZ

Carl M. O'Brien

The Numerical Algorithms Group Ltd.
NAG Central Office, Mayfield House,
256 Banbury Road, Oxford, 0X2 7DE

ABSTRACT

This paper describes advances in the development of a know ledge-based front-end for the statistics package GLIM 3.77, employing logic programming methods and tools, including sigma-PROLOG and APES. The domain of application - computer-based statistical analysis - requires that a changing knowledge base be handled by the system, GLIMPSE*. The developing system is suitable for users with different levels of expertise, including those who wish to act independently of the system' s advice. Two features are incorporated into GLIMPSE that are suitable for other front-ends and expert systems

- a facility for suggesting answers to a user who requires guidance;
- an evaluator that suspends when information is missing and suggests an action to find the missing information.

Previous and current methods of providing advice are discussed.

I Introduction.

GLIMPSE is a system being developed as a front-end to the statistics package GLIM 3.77 (Payne 1985). It is designed to provide semantic and syntactic help, and advice on generalized linear modelling and statistical strategy. The domain - computerized statistical analysis - means that the system must deal with a changing knowledge base due to the discovery of new facts gleaned during an analysis. It must also be suitable for users with different levels of expertise, including those who wish to act independently. We describe the features that distinguish GLIMPSE from other front-ends and suggest a suitable methodology for providing advice.

We describe two features of GLIMPSE that are suitable for dealing with such requirements and that may prove useful for other front-ends and expert systems, namely

- an evaluator that suspends when Information is missing and suggests the action required to obtain it; and
- a facility for suggesting answers to a user who needs help to answer a question posed by the system.

• Development by Depts. of Computing & Mathematics, Imperial College and Numerical Algorithms Group Ltd., with funding support from the Alvey Directorate under contract ALV/PRJ/IKBS/033.

II The *Quest* of the Statistical Analyst

Computerized statistical analysis is analogous to an adventure game in which the adventurer, or analyst, is on a quest to discover enough facts about a data set to be able to make certain inferences; e.g. to decide that one or more models adequately describe the data. The analyst sets off on this quest with numerical data and a limited set of known facts about them. The analyst has one main tool - the statistics package - to use on this quest and limited resources of time and money. He/she may use the calculation and graphics facilities of the package to gain further facts or clues to facts about the data, but each use depletes some of the available resources.

Some of the clues found may lead down blind alleys and so the retracing of one's steps may be necessary. The information found down these blind alleys may not, however, be wasted; it may prove useful later.

III Help is at hand

GLIMPSE is designed to provide advice to the analyst during this voyage of discovery. The analyst may ask for advice on how to use the command language of the underlying statistics package to carry out certain actions and, more importantly, for advice on modelling and statistical strategy. In the latter case, the analyst is essentially asking what action should be carried out given the facts currently known and clues found. If the action suggested is carried out and more clues are found, these will be used by GLIMPSE when providing further advice. That is, GLIMPSE advises the analyst on possible interpretations of the output from the statistics package.

The requirements placed upon GLIMPSE are very exacting due, in part, to the complex nature of statistical analysis and also to adherence to certain principles. These are described below and compared with those of other front-ends:

- it must deal with a changing knowledge base. As mentioned earlier, use of the statistics package produces new facts and clues. Due account must be taken of these as they may alter the user's perspective of the data. Most other front-ends, such as SACON (Bennett et al, 1978), ECO (Uschold et al, 1984) and PIP (Pauker et al, 1976) assume a static knowledge base - the user is not assumed to gain additional knowledge during use;

- the user should be free either to act independently or to reject advice given. GLIMPSE should take account of any information gained independently by the user when giving advice at later stages. Most other front-ends to statistical packages, such as REX (Gale, 1986), either take no account of the results of independent actions or constrain the user to follow the advice given;
- it must be suitable for users with different requirements and different levels of skill or expertise.

IV Suggested answers - a method of layering advice.

The front-end is based upon the logic-based expert system shell, APES (Hammond and Sergot, 1984), which is written in sigma-PROLOG (McCabe et al, 1984). Two key features of APES are the provision of rule-based explanations - how, why and why not - and declarative dialogue. Declarative dialogue is based on the principle of Query-the-User (Sergot, 1983) which, in its simplest form, involves asking the user to supply answers, or solutions, to relationships not defined by rules in the data base. An essential feature of APES is that it keeps a trace of the computation, used as a basis for explanations.

When asked a question the user is normally expected to supply the information requested directly. However, a recent extension means that, where appropriate, the user may, when asked a question, decline to answer directly and, instead, ask for a suggested answer from the system. The user may adopt any suggestion as the basis for his/her answer or may reject it entirely. This feature is particularly useful where the user is expected to have deeper knowledge than the system. For example, we assume the user to be better than the system at interpreting graphical output, but are willing, if requested, to suggest an interpretation based on certain summary statistics. The behaviour is demonstrated in the following specimen dialogue (user's responses underlined).

Is the graph of (ht vs age) linear? <u>SUGGESTION</u>
==> Suggested answer is 'yes'. <u>OK? WHY</u>
1. curvature statistic for (ht vs age) = 0.6
2. linearity indicated if curvature < 2.0
3. 0.6 < 2.0

Because GLIMPSE acknowledges the user's superior expertise, it is happy to accept an answer 'no' despite the affirmative suggestion. In other words, a curvature statistic < 2 is not assumed to imply linearity, merely to indicate it. This approaoh differs from that taken in other systems, such as ONCOCIN (Shortliffe et al, 1981), where, if the user is unable to supply a particular value, a default value may be automatically used instead. We prefer to suggest a response and leave the final decision with the user.

This feature allows us to layer advice. The user who is more experienced at interpreting graphs may answer directly whilst the less experienced user may request further help. The help given may itself be in terms of another relation for which a suggested answer may be requested, which leads to the possibility of multi-layered advioe. As well as helping the user it is also of use to the knowledge engineer, providing the potential to increase the expertise of a system without altering higher-level rules.

V. Early attempts to provide advice

When asked for advice GLIMPSE responds with an advisable action, or command, following any necessary dialogue. Early versions made extensive use of the 'suggested-answer' feature to provide advice.

To demonstrate this, we will use the advice provided for a model selection process. In outline, model selection is a process of examining a set of 'free terms' in order to determine whether each of these terms is either significant, i.e. should be included in a model, or insignificant, i.e. should be excluded. It is an iterative process and at any point a term may be included, rejected or left as doubtful. As more terms are either included or eliminated from consideration, the number of free terms is gradually reduoed. The aim is to be left with no free terms. In the following example dialogue, the user's response is again shown underlined.

Advice: examine terms (x1 x2 x3 x4 x5 x6) OK
. . . .{system output from command}
Which terms should be included? <u>SUGGESTION</u>
= = > I would suggest : x1 x2. OK? <u>YES</u>
Which terms should be rejected? <u>SUGGESTION</u>
= = > I would suggest : x3 OK? <u>NO</u>
I repeat,
Which terms should be rejected? x4 .x5
Advice: examine terms (x3 x6). . . . .and so on.

There were two problems with this approach:

- for the user wishing to accept the system's suggestions the process was tedious;
- if a user rejected a suggestion, could the resulting action be truly described as advioe, and could the system be said to be advising a strategy if suggested answers were rejected?

VI Current approach to providing advice

The root cause of the problems in the earlier approach was that the two distinct functions of providing help with the command language and providing guidance on statistical strategy were merged through mis-use of the 'suggested-answer' feature. So that the user could intervene and influence the command 'advised', the user was asked to supply oertain parameters even in situations where the strategy employed by the system dictated the parameters that should be used. Only if the user asked for, and accepted, a suggestion was the system strategy truly followed.

The approach now taken separates these two functions:

- the user may request help with the command language. A dialogue ensues from which the system determines the action the user wishes to perform and then advises an appropriate command;
- the user may request statistical help. The system suggests an action based on its own strategy. The user is queried only if the system does not have the information necessary to come to a conclusion. The suggested-answer facility is used only in situations where the system has rules available but considers the user better able to provide the answer.

Thus, in the example given above, the user is no longer asked about inclusions or exclusions as the system forms its own conclusions based on its built-in strategy. The revised interaction might be as follows (user' s input underlined):

Advice: examine terms (x1 x2 x3 xM x5 x6) ok
. . . . . . {system output from command}
Advice: examine terms (x4 x5 x6) ok
. . . . . . {system output from command}
Advice: examine terms (x4) STOP

As with the last advised command, the user is still free to reject any action advised.

VII Suspending evaluator for jailing information

The statistical advice given is derived from logic rules of the form

advised-action (_action) if {..conditions}

where underscore "_M indicates a variable.

If we wish to advise the user to carry out the sequence of individual actions making up a procedure, we may describe the procedure by a set of logic rules of this form. For example, during the model selection process we need to advise the user to set a parameter known as the 'baseline mean deviance* (bmd) but may need to estimate the value of this by finding the mean deviance of a statistical model. We therefore wish to advise the user to:

1. find the mean deviance,
2. set the bmd to the mean deviance found.

The following logic rules describe this:

1. advised-action ((find the mean deviance)) if
   an-estimate-of-bmd-is-required &
   not established-mean-deviance (_x)
2. advised-action ((set the bmd to be _x)) if
   an-estiamte-of-bmd-ls-required &
   established-mean-deviance (_x)

Examination of these rules leads to the following observations.

a) The first rule suggests an action to find out information, whilst the seoond suggests an action to set a variable; i.e. change a system variable based on the information found by carrying out the first action.
b) The conditions of the rules differ only in that in the first rule the information has not been found, whilst, in the seoond, it has.

In order to be able to advise what value a system variable should be set to, we may need to advise the user to elicit many items of knowledge. In general, we require a rule of the form

advised-action ((find _fact)) if
    {_some-oonditions} &
    not established-this-particular (_fact)

for each fact to be found, as well as a rule advising the resulting 'setting' action based on these facts.

Although this is a feasible method of representing our knowledge about the procedure, it results in a set of similar, repetitive rules, differing only according to whether certain faots are known or not. Evaluation of these numerous rules is naturally inefficient.

To overcome these problems, we use a new evaluator that can not only fall or succeed a goal atom, but also *suspend*. We declare certain relations to be *suspendable* and, at the meta-lev el, define

- conditions that must be satisfied for these relations to be solvable (either success or failure); and
- actions that should be advised if these conditions are not satisfied and suspension occurs.

If a goal suspends because the conditions are not satisfied, evaluation is interrupted and the evaluator looks up the advisable actions and suggests these to the user. We emphasize here that once the advised actions have been carried out, evaluation does not simply continue from where suspension occurred. This is not possible as actions may have changed both the system* s and the user's knowledge so that previous conclusions may no longer be valid.

As an example of this method, we may replace the two object-level rules defining advisable actions given earlier by a single object-level rule and additional meta-level assertions.

Object-level:

advised-action ((set the bmd to be _x)) if
    an-estimate-of-boid-is-required &
    mean-deviance (_x)

Meta-level:

```
suspendable (mean-deviance)
conditions-to-be-solvable (mean-deviance (_x)
          (established-mean-deviance (_x)))
actions-to-solve (mean-deviance (_x)
          (find the mean deviance))
```

This approach has the advantages of:

- fewer object-level rules, so more efficient computation;
- improved rule-based explanations for advising a 'finding' type of action. Previously, explanations would have been top-down, based simply on the rule(s) used in the proof, for example:

```
advised-action ((find the mean deviance))
because an-estimate-of-bmd-is-required &
        not established-mean-deviance (_x)
```

Now, they may be bottom-up, based on the missing information and its consequences, i.e. the 'setting' action being worked towards:

```
the action (find the mean deviance)
will enable us to solve the goal
     mean-deviance (_x)

if   mean-deviance (_x)
then advised-action ((set the bmd to be _x))
```

VIII Combining suspending evaluator and Suggested answers

Use of the suspending evaluator proves invaluable when combined with the suggested-answer feature. We described earlier how suggested answers may be used to provide subsidiary advice. Now, if the user asks for a suggested answer to a 'yes/no' type of question, the suggestion he/she expects to receive is either 'yes' or 'no'. However, the system may not be able to make such a suggestion if inadequate information is available. By using the suspending evaluator to interpret the subsidiary advice, we may suspend the evaluation and use the reason for suspension as a basis for advising an action, rather than suggesting a 'yes/no' answer. For example, consider the earlier example about curvature. If the curvature statistic was not known, then, rather than suggest 'yes' or 'no' to the question about linearity, the evaluator could suspend and propose to the user that the action

find the curvature statistic for (ht vs age)

be carried out.

IX Conclusion

Front ends to statistical (and other) analysis packages involve advising a user to carry out a prooess of information gathering. Use of an evaluator that suspends when information is lacking, in combination with a suggested-answer facility, permits us to provide advioe on actions to be carried out in a way that is efficient and suitable for users with differing levels of expertise.

ACKNOWLEDGEMENTS

The authors would like to thank Peter Hammond and Jonn Nelder for their comments on earlier drafts of this paper.

REFERENCES

Bennett, J. S., Creary, L., Engelmore, R. S., Melosh, R. , 1978. SACON: A know ledge-based consultant for structural analysis. Stanford Computer Science Report CS-78-699.

Gale, W.A. , 1986. REX Review. In Artificial Intelligence & Statistics (ed. W.A. Gale), pp 173 - 227, Addison Wesley.

Hammond, P. and Sergot, M. J. , 1984. APES: Augmented Prolog for Expert Systems, Reference Manual. Logic Based Systems, Surrey.

McCabe, F. G., Clark, K. L., Steel, B. D. and French, P.D. , 1984. sigma-PROLOG 1.0 Programmer's Reference Manual. LPA Ltd., London.

Pauker, S. G. , Gorry, G. A. , Kassirer, J. P. and Schwartz, W. S. , 1976. Toward the simulation of clinical cognition: taking a present illness by computer. The American Journal of Medicine, vol. 60, pp 981-995.

The GLIM System, Release 3.77, Generalised Linear Interactive Modelling Manual (ed. C. D. Payne), 1985. Numerical Algorithms Group, Oxford.

Sergot, M.J. , 1983. A Query-the-User facility for logic programming. Integrated Interactive Computer Systems (eds. P. Degano and E. Sandwell), North-Holland.

Shortliffe, E. H. , Scott, A. C. , Bischoff, M.B. , Campbell, A. B., Van Melle, W., and Jacobs, C.D., 1981. 0NC0CIN: An expert system for oncology protocol management. Procs. UCAI-1981, pp 876 - 881.

Uschold, M. , Harding, N. , Muetzelfeldt, R. and Bundy, A., 1984. An intelligent front end for ecological modelling. Research Paper 223, Dept. of Artificial Intelligence, University of Edinburgh.

Author Index