Optimizing the Costs and Benefits of Natural Language Generation

Ehud Reiter*and Chris Mellish†
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
BRITAIN

Abstract

We discuss the costs and benefits of using natural-language (NL) generation technology to produce documentation and on-line help messages, and propose some 'intermediate' generation techniques that provide many (although not all) of the benefits of more principled 'deep' generation techniques but at a significantly lower cost.

1 Introduction

Most research on natural-language (NL) generation has stressed what might be called 'deep' techniques, where text is generated in a principled way from comprehensive knowledge bases that describe the domain, the user, and the context. In many circumstances, however, constructing such knowledge bases can be prohibitively expensive, and the benefits gained by using deep NL generation techniques may therefore be outweighed by the cost of constructing the necessary knowledge bases. There are also situations where constructing the text in a principled way would require a prohibitive amount of computer time; and situations where not enough is known about a particular generation task to enable a principled generation system to be built.

One alternative to 'deep' generation is to use what might be called 'intermediate' techniques, which sacrifice some of the benefits of deep generation in order to lower the cost of creating the necessary knowledge bases, reduce the amount of computation time required, or work around poorly understood areas of the generation process. Intermediate techniques can be characterized by listing, on the one hand, what benefits of deep generation are foregone by using the intermediate techniques, and what domain or environmental constraints must be met in order for the techniques to be applicable; and on the other hand, what improvements they bring over deep generation in terms of reduced authoring cost, reduced computation requirements, etc. Whether a particular intermediate technique is useful in a particular application then depends on how closely the costs and benefits of the technique match the goals and resources of the application.

The goal of this paper is to present a general cost–benefit framework which can be used to analyze intermediate techniques, and to illustrate this framework by discussing and analyzing some of the specific techniques used in the IDAS NL generation system being developed at Edinburgh [Reiter *et al.*, 1992]. Our hope is that this will encourage other researchers and developers to similarly describe the techniques they have developed in terms of costs and benefits; perhaps at some stage it will then be possible to construct a catalogue of different techniques, with their associated costs and benefits, that application builders will be able to consult.

The discussion of costs and benefits in this paper will be qualitative, not quantitative. It is difficult (and expensive) to measure the types of costs and benefits we are most interested in (e.g., the time and effort required to create the relevant knowledge bases, or the total lifecycle cost of maintaining a document) in a realistic and controlled manner. The only numerical data that we have collected so far comes from user-effectiveness trials, and while user-effectiveness is important (and the data from our trials is encouraging), it is only a small part of the overall cost/benefit analysis. The qualitative framework and analysis given here is in some sense less desirable than a rigorous quantitative analysis, but we believe that it can still be useful as a way of presenting and discussing some of the engineering issues involved in building NL generation systems.

2 Costs and Benefits of NL Generation

In this section we examine some of the main costs and benefits of using natural-language generation to produce documentation or help messages from a knowledge base (KB). This is the goal of IDAS, and has also been the goal of many other projects, including COMET [McKeown *et al.*, 1990], WIP [Wahlster *et al.*, 1991], EUROHELP [Breuker, 1990], TAILOR [Paris, 1988], and JOYCE [Ram-

*E-mail address is E.Reiter@ed.ac.uk.
†E-mail address is C.Mellish@ed.ac.uk.

bow and Korelsky, 1992].

2.1 Benefits

Some of the benefits that natural-language generation potentially offers over canned documentation and help messages are:

Contextual Tailoring: The output texts can be tailored to the user's expertise level [Paris, 1988; McCoy, 1988], the task she is attempting to perform [Wilensky *et al.*, 1988; Breuker, 1990], and the current discourse context [McKeown, 1985]; tailoring can be done on both the informational content of a text and on the way this information is expressed in language [Reiter *et al.*, 1992]. The benefits of tailoring are perhaps most obvious for help systems, but it also would be very useful if the different types of documents needed for a complex system (user manual, maintenance guide, design description, etc.) could be generated from a single integrated domain KB.

Multimodal documents: Generation systems can automatically produce pictures as well as texts from a domain knowledge base [Feiner and McKeown, 1990; Wahlster *et al.*, 1991], and can embed mousable hypertext links in the textual portion of a message [Stock, 1991; Reiter *et al.*, 1992].

Multilingual Generation: An NL generation system can in principle generate texts in multiple languages from the same domain knowledge base [Hovy *et al.*, 1992]. IDAS, for example, can generate texts in French as well as English, if the domain knowledge base has been created in a language-independent manner [Bellos, 1992].

Maintainability: NL generation technology can make it significantly easier to maintain documentation as the machine or program being documented is upgraded or released in new configurations.

Enhanced maintainability is of great interest to industrialists, but has not been much discussed in the academic literature, so it is perhaps worth elaborating on this. As with software, the biggest portion of a document's life-cycle costs is maintenance, not initial creation. This is especially true when the document describes a machine or program that is being constantly upgraded to enhance its functionality and use new technology, or is being sold in an ever-increasing number of different configurations. Updating paper manuals to be consistent with upgrades or new configurations is an expensive and time-consuming task, largely because the changes need to be made in many places in many documents. In some cases, however, it is possible to express an upgrade or configuration change fairly concisely in a domain knowledge base, and NL generation then allows the relevant documentation to be automatically modified to take account of these changes. For example, if the name of a command is changed in a program, it is easier to make this change once in a domain knowledge base and have it automatically propagated to all relevant pieces of documentation, than to search for and manually modify every place in the documentation where the command is mentioned.

Enforcement of Style and Content Rules: NL Generation systems can generate text that is guaranteed to meet externally imposed writing rules (e.g., AECMA Simplified English [AECMA, 1986]) and content requirements (e.g., the UK Army Equipment Support Publications rules). Guaranteed conformance to such rules is again a topic that has not been discussed much in the academic literature, but that seems to be of interest to many industrialists.

Automatic Document Creation: Many documents are essentially just design descriptions, and a significant proportion of these documents could in principle be automatically produced from an appropriate CAD (Computer-Aided Design) database, especially if these documents are intended to be read by domain experts, not by naive end-users. The JOYCE system [Rambow and Korelsky, 1992] represents a step in this direction.

Contextual tailoring and multimodal document generation are beneficial because they make documentation more useful and effective; multilingual generation, enforcement of style/content rules, enhanced maintainability, and automatic document creation are beneficial because they reduce the total life-cycle cost of a document (i.e., editing for conformance to style and content rules, translation, and maintenance as well as initial creation). In our experience, many, perhaps most, academic researchers focus on benefits of the first kind, i.e., they view NL generation as a way of improving document quality. Industrialists, on the other hand, are often most interested in benefits of the second kind, i.e., they are looking for ways to cut the high costs of producing, maintaining, and translating documentation, and are interested in NLG techniques in so far as they provide a way to do this.

2.2 Costs

In our experience in the IDAS project, the single biggest cost of producing documentation with NL generation techniques is setting up the requisite knowledge bases for the generation system to generate text from. Creating a domain knowledge base generally requires more work than writing canned-text documentation or help messages (unless it is possible to automatically derive part of the KB from a CAD system), and the generation approach also requires user, linguistic, and contextual models. Thus, generating documentation and help mes-

sages from a knowledge base is only useful from a practical perspective if (a) creating the necessary knowledge bases is not much more difficult than entering canned-text documentation and (b) the generation system offers significant benefits over canned-text documentation.

In some cases the computational cost of generating documentation or help messages may also be a significant factor; on-line help systems, in particular, are not of much use unless they can produce responses within a few seconds. The effort to write the generation software must also be taken into account, although this hopefully can be amortized over many applications.

A perhaps related problem with generating text from a knowledge base is that the state-of-the-art in NL generation is such that it is not known how to perform certain generation tasks in an adequate way, even given sufficient resources in terms of domain knowledge and computation time. For example, although a substantial amount of research has been done on the problem of combining clauses into coherent sentences and sentences into coherent paragraphs, very little work has been done on combining paragraphs into coherent sections, chapters, and documents, and thus not enough is known about this problem to make it possible to construct higher-order structures in a principled way (unless the document in question is expected to have a very stereotypical format and content).

The cost of building the necessary knowledge bases, and performing the necessary computations, is a problem in many NL (and AI) applications, not just generating documents from a knowledge base. In the machine translation field, for example, the TAUM-AVIATION followup project to the very successful METEO system was canceled because an analysis showed that the necessary domain and linguistic knowledge bases would be too expensive to construct in TAUM-AVIATION's domain of aircraft manuals [Isabelle and Bourbeau, 1985]. Creating appropriate domain and linguistic models can also be a significant problem for users of NL database interfaces [Walker *et al.*, 1992].

The costs of using NL generation will probably decline with time. Computers will almost certainly continue to get faster and cheaper, and the research frontier will undoubtably expand to cover many of the areas that are poorly understood today. The per-application cost of building the necessary linguistic knowledge bases should decrease as linguistic KB's are built that can be shared by many applications; indeed, shared grammars and lexicons are already beginning to appear, such as the ALVEY toolkit [Grover *et al.*, 1989]. The increasing computerization of the design process should also make it more feasible to automatically derive part of the documentation KB from CAD databases. Eventually it should be possible to integrate documentation KB building into the design process, instead of leaving it as a separate task; this will reduce the effort required to construct the documentation KB, and may have the added advantage of helping the designer detect design errors at a relatively early stage. These developments will make it more practical to use deep techniques in generation systems, or at least 'deeper' intermediate techniques.

3 Intermediate Approaches in IDAS

The IDAS system generates on-line documentation and help messages for users of complex physical machinery, using a domain knowledge base constructed for this purpose and user-task, user-expertise, contextual, and linguistic models. The system being documented in the first version of IDAS is an ATE (Automatic Test Equipment), a complex device made by Racal Instruments to test potentially faulty electronic devices.

From a benefit perspective, IDAS's goals include enhanced maintainability, automatic creation of hypertext links, and some degree of user tailoring. Multilingual generation, obeying stylistic or content rules, and automatic document generation from a CAD database are not primary goals of IDAS. From a cost perspective, one of IDAS's goals is to allow the necessary KB's to be built primarily by domain experts or technical writers, with as little intervention by AI experts as possible; hence KB authorability is a primary concern. Computational efficiency is also of some importance, since IDAS will be used to generate texts in an interactive environment.

In this section we shall examine two intermediate techniques used in IDAS to decrease authoring costs, namely hybrid action representations and rule-based content determination. Our goal is not just to present these particular techniques, but also to show how intermediate techniques can be analyzed and justified from a cost-benefit perspective.

In addition to the above techniques which are aimed at reducing authoring costs, IDAS also uses intermediate techniques to reduce computation time and 'work around' poorly understood NL generation tasks. Perhaps the best example of a technique that reduces computation time is IDAS's referring-expression generation model, which is described in detail in [Reiter and Dale, 1992]. This model was largely motivately by a computational analysis that showed that some of the models proposed for generating referring-expressions in previous work made the generation task NP-Hard; one of the most interesting things about it is that the 'fast and simple' algorithm developed for generating referring expression in IDAS seems to be a much closer approximation to what human speakers do than some of the more complex algorithms proposed in previous research. But perhaps this should not be a surprise; evolution, after all, is all about finding engineering solutions that maximize cost-benefit tradeoffs in particular environments, so it is perhaps to be expected that human speakers might also use 'intermediate' techniques in some circumstances.

A good example of an intermediate technique whose

aim is to sidestep a poorly-understood NLG task is IDAS's use of hypertext mechanisms to obviate the need to perform high-level text structuring (e.g., assembling paragraphs into documents); this work is discussed in more detail in [Reiter *et al.*, 1992] and [Levine *et al.*, 1991]. The basic idea is to use hypertext mechanisms to enable users to dynamically select the paragraphs they wish to read, and therefore in essence perform their own high-level text-planning. In other words, IDAS avoids the need to perform high-level text-structuring by getting the user to do this task.

3.1 Hybrid Action Representations

One of IDAS's tasks is to tell the user how to perform actions. IDAS can perform this in a 'deep' manner by generating text from a case-frame representation of the action to be performed; it is, however, impractical to expect domain experts or technical authors, who in general have minimal experience with AI techniques, to create such case frames by hand. Such authors find writing text easier than building case frames, so the ideal solution would be to have the authors write text and then convert this text into case frames with an NL understanding system. Given the state of the art in NL understanding, it is difficult to reliably and unambiguously translate arbitrary texts into IDAS's internal case-frame notation, but some processing can certainly be done. This has led to the notion of 'hybrid' action representations which mix proper knowledge-base structures with canned-text fragments; the former represent pieces of the input text that the analysis system can confidently analyze, while the latter are used for unanalyzable portions of the text.

More specifically, we support two hybrid representations: *canned text with embedded KB references* (EKR), and *case frames with textual case fillers* (TCF). In the EKR representation, references to machine components and other KB entities can be embedded in a canned-text action representation; the generation system then generates appropriate referring expressions for these references when it processes the EKR form. In the TCF representation, the IDAS case-frame representation is used, but case fillers are allowed to be canned text; these are then inserted into the generated sentences in appropriate positions. Examples of these representations are:

Canned-text: *Remove any connections to the board*

EKR: *Carefully slide [*Board*] out along its guides*

TCF: REMOVE(actor=User, actee=Board,
source=Instrument-Rack, manner="*gently*")

Case-frame: PUT(actor=User, actee=Board,
destination=Faulty-Board-Tray)

EKR representations can be created by scanning a text file and searching for references to domain entities; TCF representations are constructed by running the input text through a simple parser, and leaving any portion of the text that cannot be unambiguously processed as a canned-text filler. Together with our industrial collaborators, we have written authoring tools that perform both of these tasks (e.g., [Marshall, 1992]), with some support for graphical entry of information as well. The EKR authoring tool can process all texts (and simply returns pure canned-text if it cannot detect any comprehensible references to domain entities), while the TCF tool cannot at present deal with some constructions, such as conjunctions (e.g., it could not process *Remove and clean the plate*). Both tools greatly reduce the authoring cost of actions, since they can now be entered in textual form by domain experts and technical authors.

With regard to the benefits described in Section 2.1, both EKR and TCF allow some tailoring to be done according to the user and discourse model, particularly with regard to the ways in which objects are described. For example, under either representation a reference to DMM-Board can be realized as *the DMM*, *the digital multimeter*, or *the board in slot 6 of the instrument rack*, depending on the user's expertise and the discourse context. The TCF representation also allows some tailoring to be done with verb lexicalizations, such as the generation of *Buzz out the cable* instead of *Test the cable with the connectivity checker* (*buzz out* is a term used by maintenance experts for the action of testing with a connectivity tester). Both EKR and TCF also allow hypertext links to be automatically added from component mentions to more detailed descriptions of the components; this is a multimodal benefit of generation. Maintainability is supported primarily by allowing actions to be specified at a high-level in the object IS-A hierarchy and inherited by lower objects; for example, a How-To-Remove-Action can be stated for the general class VXI-Chassis-Board, and inherited by specific boards including the digital multimeter, the counter-timer, the frequency generator, etc.

On the other hand, EKR and TCF do *not* support either multilingual generation or strict enforcement of style rules; if these benefits are important in the application, then EKR and TCF should not be used.

In summary, while the 'deep' case-frame representation has the most benefits, it is unrealistic to expect technical authors to directly create such representations, and natural language understanding technology is probably not capable of reliably and unambiguously generating case-frames from human-authored texts. It is, however, fairly straightforward to do some processing of human-authored texts and produce the EKR or TCF intermediate forms; this course of action has a much lower cost than requiring authors to create case-frames, but also loses some of the potential benefits of a case-frame representation (e.g., multilingual generation). The intermediate forms do have significant benefits over pure

canned-text documentation, however, since they allow the generation system to do some user and contextual tailoring, and also allow it to automatically add hypertext followup links to component mentions. EKR and TCF thus provide substantial benefits over canned text at a much lower authoring cost than required by case-frames, and may be appropriate in applications such as IDAS that do not emphasize multilingual issues, or the need to obey stylistic rules.

3.2 Rule-Based Content Determination

Content determination is the problem of determining what information should be communicated in order to adequately respond to the user's queries. Much research has been done on using planning techniques to accomplish this task; EUROHELP [Breuker, 1990], for example, first infers the user's goal by using plan-recognition techniques, and then creates a plan to satisfy this goal. Unfortunately, while planning-based content-determination techniques permit responses to be precisely tailored towards the user's task and goals, they also have a very high cost, since building a comprehensive plan library that supports planning and plan recognition in a realistically-sized domain is a major endeavour, and one that probably must be done by AI experts, not domain experts or technical authors. Planning and plan recognition can also be extremely expensive in computational terms unless heuristics are used to control the plan search process (as they are in EUROHELP); these heuristics can limit the generality and robustness of the planning and plan recognition process, however, and hence make the systems less likely to respond appropriately in unusual situations, which is one of the main benefits of using the planning approach in the first place.

Given IDAS's authorability constraints, and also worries about the required amount of computation time, the decision was made to go for a less ambitious 'intermediate' approach based on content-determination rules. Essentially, the domain KB author enters a series of rules that say 'if the user asks question Q about a component of type C in the context of task T, then convey to him the following information about the component'; an example of a content rule is 'if the user asks a WHAT question about a MACHINE in the context of an OPERATIONS task, inform him of the machine's manufacturer, model number, and colour'. There are a limited number of questions and tasks[1] in IDAS, and inheritance can be used to define the applicability of a rule (i.e., a rule can apply to all Maintenance tasks or only to the Repair-Part task, and to all machines or just to instruments), so the number of rules that need to be authored is not over-large. Given the relatively small number of rules, it may be reasonable to expect them to be created by specialists with some AI expertise; one of the IDAS authoring tools is also being modified to allow normal authors to enter this information with a menu-based interface.

The rule-based content-determination system has a much lower authoring cost than a plan-based system. It also has a lower computational cost, since the system simply has to find and execute the rule that matches the current situation; there is no need to perform the substantial amounts of inferencing required by both planning and plan-recognition.

With regard to the benefits of Section 2.1, rule-based content determination allows some amount of tailoring according to the user's task, but not as much as is allowed by the plan-based approach (unless there is a large number of rules and tasks, which leads to authoring problems). It does allow for the enforcement of externally imposed content requirements, which exist in some domains (e.g., to meet IEEE standards), and also supports multilingual output, since the rules are phrased at a conceptual level. Perhaps most significantly for IDAS, it supports enhanced maintainability; if the machine being documented is upgraded or reconfigured by changing one of its components (e.g., installing a more sensitive digital multimeter in the ATE), the documentation maintainer can simply enter the specifications for the new component, and the relevant IDAS output texts will automatically be updated. The rule-based approach may also support some degree of automatic document creation, if the documents in question follow rigid formats, but we have not investigated this in detail.

The rule-based approach will only work in some domains, however. In particular, it works best when the number of tasks the user might potentially perform is small and fairly predictable; a large number of tasks would make rule authoring more expensive, and it also would be difficult to build an appropriate set of rules if it was not known ahead of time what tasks the user was likely to want to perform. Another important point is that an IDAS user can always ask followup or clarification questions if she is unsatisfied with the response to her query, which means that the cost of generating an occasional inappropriate response is fairly low; rule-based techniques might be less appropriate in applications where even infrequent inappropriate responses were unacceptable.

Rule-based content-determination thus has a much lower authoring and computational cost than plan-based techniques, at the price of producing responses that are not as well tailored to the user's task and other contextual parameters; the technique also requires a domain with a limited number of tasks, and an interactive medium that allows followup questions. If the domain and medium is appropriate, and the loss of tailorability is acceptable, rule-based content determination can be a valuable technique for the application builder.

[1] IDAS does not use plan recognition techniques to infer the user's task; it simply asks her (when she starts the system) which task she will be performing.

Potential Benefit	Hybrid Action Rep	Content Rules	IDAS Goals
Tailoring	canned portion cannot be tailored	fine-grained content-tailoring is difficult	important
Multimodal	OK	OK	hypertext
Multilingual	not possible	OK	
Style/Content rules	cannot enforce style rules	OK	
Maintainability	OK	OK	important
Automatic Creation	OK	OK	
Domain and Environment Restrictions:			
Number of tasks	no restriction	should be small	small
Interaction type	no restriction	better if interactive	interactive

Table 1: What Benefits are Given up with Intermediate Techniques?

Table 1 summarizes which of the potential NL Generation benefits mentioned in Section 2.1 are preserved by the intermediate approaches described above. We have shown IDAS's goals and domain parameters in the table, and the reader can see that there is a reasonably good match between them and the situations where hybrid action representation and rule-based content determination are most useful; this is not surprising, since IDAS's requirements motivated our search for appropriate intermediate generation techniques. Researchers and developers who are working on applications with different goals and domains will no doubt find that other intermediate techniques are most useful for them.

4 Conclusion

Natural-language generation needs to be analyzed from an engineering perspective if it is to get out of the research lab and into real applications. This requires evaluating the costs and benefits of different techniques for performing generation tasks, and in many situations choosing techniques that are 'sub-optimal' in principle, but offer the best cost-benefit tradeoff for the application under consideration. We have presented in this paper some 'intermediate' (i.e., somewhere in between simple templates and principled deep generation) techniques that we have found useful in our IDAS project. Our hope is that these techniques may in themselves be of some interest to application developers, and, perhaps more importantly, that our cost-benefit analysis of these techniques can be used as a model for similar analyses of other generation techniques.

Acknowledgements

The IDAS project is partially funded by UK SERC grant GR/F/36750 and UK DTI grant IED 4/1/1072, and we are grateful to SERC and DTI for their support of this work. We would like to thank the IDAS industrial collaborators — Inference Europe, Ltd.; Racal Instruments, Ltd.; and Racal Research, Ltd. — for all the help they have given us in performing this research. We would also like to thank the many other industrialists who have discussed natural-language generation with us.

References

[AECMA, 1986] AECMA. A guide for the preparation of aircraft maintenance documentation in the international aerospace maintenance language, 1986. Available from BDC Publishing Services, Slack Lane, Derby, UK.

[Bellos, 1992] Ilona Bellos. Towards a multilingual IDAS. Master's thesis, University of Edinburgh, Department of Artificial Intelligence, 1992.

[Breuker, 1990] Joost Breuker, editor. *EUROHELP: Developing Intelligent Help Systems.* European Commission, 1990. Final report on the P280 ESPRIT project EUROHELP.

[Feiner and McKeown, 1990] Steve Feiner and Kathleen McKeown. Coordinating text and graphics in explanation generation. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-1990)*, volume 1, pages 442–449, 1990.

[Grover *et al.*, 1989] Claire Grover, Ted Briscoe, John Carroll, and Bran Boguraev. The ALVEY natural language tools grammar. Technical Report 162, University of Cambridge Computer Laboratory, Cambridge, UK, 1989.

[Hovy *et al.*, 1992] Eduard Hovy, Richard Kittredge, Christian Matthiessen, Sergei Nirenburg, and Dietmar Rösner. Multilinguality and generation. In R. Dale et al., editors, *Aspects of Automated Natural Language Generation: Proceedings of the Sixth International Natural Language Generation Workshop*. Springer-Verlag, 1992. Number 587 in Lecture Notes in Artificial Intelligence.

[Isabelle and Bourbeau, 1985] Pierre Isabelle and Laurent Bourbeau. TAUM-AVIATION: Its technical features and some experimental results. *Computational Linguistics*, 11:18–27, 1985.

[Levine *et al.*, 1991] John Levine, Alison Cawsey, Chris Mellish, Lawrence Poynter, Ehud Reiter, Paul Tyson, and John Walker. IDAS: Combining hypertext and natural language generation. In *Proceedings of the Third European Workshop on Natural Language Generation*, pages 55–62, Innsbruck, Austria, 1991.

[Marshall, 1992] Sam Marshall. The IDAS action authoring tool. Master's thesis, University of Edinburgh, Department of Artificial Intelligence, 1992.

[McCoy, 1988] Kathleen McCoy. Reasoning on a highlighted user model to respond to misconceptions. *Computational Linguistics*, 14(3):52–63, 1988.

[McKeown *et al.*, 1990] Kathleen McKeown, Michael Elhadad, Yumiko Fukumoto, Jong Lim, Christine Lombardi, Jacques Robin, and Frank Smadja. Natural language generation in COMET. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, pages 103–139. Academic Press, London, 1990.

[McKeown, 1985] Kathleen McKeown. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27:1–42, 1985.

[Paris, 1988] Cecile Paris. Tailoring object descriptions to the user's level of expertise. *Computational Linguistics*, 14(3):64–78, 1988.

[Rambow and Korelsky, 1992] Owen Rambow and Tanya Korelsky. Applied text generation. In *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP-1992)*, pages 40–47, 1992.

[Reiter and Dale, 1992] Ehud Reiter and Robert Dale. A fast algorithm for the generation of referring expressions. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-1992)*, volume 1, pages 232–238, 1992.

[Reiter *et al.*, 1992] Ehud Reiter, Chris Mellish, and John Levine. Automatic generation of on-line documentation in the IDAS project. In *Proceedings of the Third Conference on Applied Natural Language Processing (ANLP-1992)*, pages 64–71, Trento, Italy, 1992.

[Stock, 1991] Oliviero Stock. Natural language and exploration of an information space: the ALFRESCO interactive system. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-1991)*, pages 972–978, 1991.

[Wahlster *et al.*, 1991] Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, and Thomas Rist. WIP: The coordinated generation of multimodal presentations from a common representation. In Oliverio Stock, John Slack, and Andrew Ortony, editors, *Computational Theories of Communication and their Applications*. Springer-Verlag, 1991.

[Walker *et al.*, 1992] Marilyn Walker, Andrew Nelson, and Phil Stenton. A case study of natural language customisation: The practical effects of world knowledge. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-1992)*, volume 2, pages 820–826, 1992.

[Wilensky *et al.*, 1988] Robert Wilensky, David Chin, Marc Luria, James Martin, James Mayfield, and Dekai Wu. The Berkeley UNIX consultant project. *Computational Linguistics*, 14(4):35–85, 1988.