Acquiring and Using Limited User Models in NLG

Ehud Reiter, Somayajulu Sripada, and Sandra Williams
Department of Computer Science
University of Aberdeen
{ereiter,ssripada,swilliam}@csd.abdn.ac.uk

Abstract

It is a truism of NLG that good knowledge of the reader can improve the quality of generated texts, and many NLG systems have been developed that exploit detailed user models when generating texts. Unfortunately, it is very difficult in practice to obtain detailed information about users. In this paper we describe our experiences in acquiring and using limited user models for NLG in four different systems, each of which took a different approach to this issue. One general conclusion is that it is useful if imperfect user models are understandable to users or domain experts, and indeed perhaps can be directly edited by them; this agrees with recent thinking about user models in other applications such as intelligent tutoring systems (Kay, 2001).

1 Introduction

It has long been recognised that NLG systems should in principle generate texts that are targeted towards individual readers, and should use detailed models of the readers when doing so. The content of generated texts should be tailored to the reader's tasks and existing knowledge; for example, a weather forecast for a pilot landing an airplane should focus on wind and visibility at the destination airport, while a weather forecast for a farmer planting crops at a farm next to the airport should focus on temperature and precipitation. The expression (microplanning) of a generated texts should be tailored to the user's linguistic abilities and preferences; for example, a smoking-cessation letter sent to someone with an age 10 literacy level should use short sentences and simple words, while a smoking-cessation letter sent to a doctor with excellent literacy could use complex sentences and specialised medical terminology. And the realisation (for example, grammar) of a text could be tailored to a user's dialect, although this is perhaps more debatable. In other words, people are very different, and texts intended for individuals will be more effective if they can be targeted towards that individual.

In accordance with this accepted wisdom, many NLG systems and models allow detailed user models to be specified. For example, plan-based content determination (Appelt, 1985; Moore and Paris, 1993) is based on detailed models of user tasks and goals, and dialect or even ideolect grammars can be specified for realisation engines such as SURGE (Elhadad and Robin, 1997) and KPML (Bateman, 1997). Zukerman and Litman (2001) review how user models have been used in a variety of NLG and NLU systems.

Unfortunately, we are not aware of any NLG systems which actually use detailed user models with non-trivial numbers of users, probably because of the difficulty of acquiring detailed user models. Such models can of course be handcrafted for demonstration purposes for a single user performing a single task, but we are not aware

of any successful systems based on detailed user models which work for a non-trivial number of real users.

The reality of NLG today is that any system with a non-trivial number of users has imperfect information about its users. It may know something about them, but its knowledge is far from complete. This raises an important question for NLG - what is the best way to acquire and use limited and imperfect information about users? Little has been published about this topic in the NLG literature; Zukerman and Litman's (2001) review, for example, says little about this topic, other than suggesting that perhaps user models can be built up during the course of a dialogue with the user.

We have struggled with this question in the course of building several NLG systems – IDAS, STOP, SUMTIME-MOUSAM, and GIRL – and used a different approach in each of these systems. In this paper we summarise the approaches we have taken and how well they seemed to work. We certainly do not have any definitive answers, but we hope our paper will at least clarify the issue. Also, perhaps one general lesson from our work is that it is helpful if imperfect user models are understandable to users or at least to domain experts. Under some circumstances this could allow people to edit and thus directly control their model; even if this is not possible, users are likely to be more helpful in the model acquisition process if they understand how the model is going to be used.

These observations fit in with recent thinking in the general user-modelling community. Fischer (2001) acknowledges that user modelling has been less successful than originally hoped, and suggests that in part this could be due to the difficulty of creating effective user models, and problems in dealing with models that are incorrect, incomplete, and/or out-of-date. Kay (2001) suggests that user models should be *scrutable* (understandable and modifiable) to the user, because (among other things) this allows users to understand what the system is doing and to correct mistakes.

Note that while research has been done on planning under uncertainty, the focus of such work is contexts where the uncertainty is well understood; for example, an object is in one of two locations (Collins and Pryor, 1995), or a diagnostic test has a well-understood false positive and false negative rate (Haddawy et al., 1995). Under these conditions it is possible to produce plans that are optimal under some cost or effectiveness criteria. However, in our experience the level of uncertainty in user models for NLG greatly exceeds what can be handled by such approaches.

2 IDAS: User in Control

One approach to the imperfect user knowledge problem is to generate a text using whatever user knowledge is available (which may not be much), and then allow the user to request additional information, clarifications, and so forth. This of course is the approach used by human speakers in dialogues, and indeed by many computer dialogue systems.

It was also used in IDAS (Reiter et al., 1995), an NLG system developed in the early 1990s which dynamically generated hypertext technical documentation from an AI knowledge base. IDAS had a facility which allowed detailed user models to be used during the generation process; but in fact no detailed models of real users were created for IDAS. Instead, users in experimental trials of IDAS would use its hypertext links to obtain more information if they needed it (Levine and Mellish, 1995).

For example, consider the question of how much detail should be in an instructional text on how to change a flat tire on a bicycle. Should such a text use high-level instructions such as *remove the front wheel*, or should it use more detailed instructions such as *lift the front wheel's quick-release lever*? The original IDAS vision was to make this choice on the basis of a detailed user model which stated which high-level actions the user already knew how to perform, and which needed to be broken down into substeps. However, in practice it was not possible to acquire such detailed information about IDAS users. Instead, IDAS produced a guess at an appropriate instruction sequence, often based on a coarse expert/novice distinction between users, and then allowed the user to click on a text if he or she wanted more information. For example, IDAS might initially produce the high-level text *remove the front wheel*, and the user could expand this into substeps (such as *lift*

the front wheel's quick-release lever) by clicking on the original instruction.

Of course this strategy only makes sense if the user understands what questions he can ask. This requires both a good user interface and also an intuitive and easily understandable 'question space' (using IDAS's terminology) of what questions the system can answer.

Letting the user specify what he or she wants to know is not always possible; for example, it is not possible in a system which generates paper letters such as STOP, and may not be realistic in a system used by people with limited computer confidence such as GIRL. It also may sometimes be risky, for example if a user thought he knew how to remove a wheel but in fact did not. But it certainly seems to be an effective strategy in many situations, because users usually have excellent knowledge of their own goals, tasks, and expertise, much better than any computer system.

3 STOP: Ask User for Key Information

Another response to the difficulty of getting perfect user knowledge is to try to determine which knowledge about the user is most important, and then design a questionnaire or GUI to explicitly acquire this knowledge. In such cases we may need to impose a size or time-to-complete constraint on the questionnaire or GUI, based on what we think is realistic for the target user group.

This approach was used in STOP (Reiter et al., 2003), which generated personalised smoking-cessation letters. We devised a 4-page multiple-choice questionnaire for smokers based on what previous research suggested would be the most important information for the letter-tailoring process. The STOP software then used this questionnaire (which was completed on paper, and scanned into a database) as its primary information source when generating tailored smoking-cessation letters; some information was also obtained from the smoker's medical record. The user questionnaire data only effected content decisions in STOP; in principle it would have been desirable to also take user information into account when making microplanning and realisation (expression) decisions, but this was not done.

One problem we encountered was that in retrospect the information elicited by the questionnaire was perhaps not the most important information needed by the tailoring process. For example, although we asked users about their smoking habits, beliefs, and concerns, we did not directly ask them what information they would like to see in the generated letter (for example, medical information about the effects of smoking vs. practical how-to-stop advice); in hindsight we probably should have asked for this information. But this is not a flaw with the technique, it is a flaw in how we applied it.

A perhaps more basic problem is that we were limited to acquiring small amounts of well-structured information. We could not acquire a lot of information (since smokers would not spend more than 10-15 minutes on a questionnaire), and we could not acquire unstructured information such as free-text explanations of interests and goals (since such texts could not be interpreted and understood by our software). For example, one key issue in smoking advice is why previous attempts to quit have failed. Our questionnaire had seven check boxes for standard reasons such as stress or weight gain. It did not elicit detailed information which turned out to be very important to individual smokers, such as one person's frustration with hypnosis techniques, and another's promising attempt to quit being derailed by stress caused by the death of a relative.

Also, the questionnaire approach of course only works if the user understands and can answer the questions. For example, detailed medical information about the smoker's health, such as the condition of his lungs, would have been useful but in general we could not expect smokers to have this information. Another example is whether the smoker is addicted to nicotine (again important information for selecting appropriate cessation advice). Many smokers have incorrect beliefs about whether they are addicted, so instead of directly asking this question, STOP inferred addiction status from a set of questions devised by Fagerström and colleagues (Heatherton et al., 1991), such as whether the smoker smoked within 30 minutes of waking up.

In other words, in IDAS we believed that users themselves had the best knowledge of relevant in-

formation such as their tasks and goals. In STOP, however, we believed that users might not have good self-knowledge of some important information, such as addiction status.

In summary, a questionnaire can work well if we need a small amount of well-structured information, and we believe that users have (and will provide) this information. Otherwise, we should consider other approaches.

4 SumTime-Mousam: Domain Expert Creates a Model

Another approach to creating user models that usefully approximate reality is to get a domain expert (or 'knowledge engineer') to build the model. That is, the domain expert meets with users and discusses their needs and constraints, and from this develops a user model for a software system.

This approach was used in SUMTIME-MOUSAM (Sripada et al., 2002), which generates weather forecasts for offshore oil rigs; these are essentially summaries of the output of a numerical weather simulation, where the summarisation is controlled by a model of what is important to the user. Such forecasts are used by oil company staff to make specific decisions, for example on how to unload supply boats and when to schedule diving operations. If we had perfect knowledge about what decisions needed to be made and what the constraints on these decisions were (for example, what sea conditions were too rough for the particular diving equipment currently at a rig), then we could generate perfectly tailored forecasts using plan-based techniques. Unfortunately, this information is not available.

Instead, we included in SUMTIME-MOUSAM some parameters which made sense to expert meteorologists (such as what changes in wind speed were significant in different contexts), and let an expert set the parameters appropriately for different users. The expert had previously discussed with the end user what the user's tasks and needs were, and based the parameter settings on these discussions and on his expertise. These parameters essentially constituted a 'user model' which was intended to apply to an entire rig, and cover all types of operations.

We have not yet evaluated the usefulness of the generated forecasts with end users, so we do not know how effective this strategy is. However, we can make some observations. First of all, building such models requires making a tradeoff between the range of situations they cover and their effectiveness. A model which covered all possible decisions would insist that all data from the numerical simulation be communicated, and no data be summarised; this would negate the usefulness of textual summaries. On the other hand, a model that was tailored just to the most typical decisions would generate texts that were well suited to those situations, but not to others.

For example, in general light winds (less than 15 knots) have minimal impact on rig operations, so when the wind is light there is no need to report (for instance) a small change in wind direction, such as N to NNE. However, there are some unusual operations, such as flaring gas, when small changes in wind direction are relevant even with light winds. If we had perfect user knowledge we would report such changes if and only if the user was flaring gas or otherwise doing an operation for which this information was important. However, since we do not know what operations are planned for a particular day, we have to choose between either never reporting such changes (which somewhat improves forecast quality for most days by eliminating unnecessary information), or always reporting such changes (which greatly improves forecast quality on those rare days when the information is important).

One solution to this problem would be to allow users to explicitly specify information about their planned tasks and known constraints (such as whether they planned to flare gas), via a software package which was installed on the rigs. We have not seriously investigated this because of the substantial cost of developing, installing, and maintaining such a system, plus the cost of training users so that they entered the correct information at the correct time. In other words, SUMTIME-MOUSAM as it currently exists fits smoothly into current operational procedures and does not require users to install new hardware or software or change the way they work; this would not be true of a system which required users to enter information about their tasks and constraints.

Another issue with SUMTIME-MOUSAM's approach is that the parameters and model must be understandable to the domain expert. The first versions of SUMTIME-MOUSAM used fairly simple and hence understandable algorithms for data analysis and text generation, but the most recent version of the system uses more sophisticated algorithms which are less easy for someone who is not a computer scientist to understand. Hence it is harder for the domain expert to build models for the most recent version of the system than for previous versions.

Finally, economics means that domain experts cannot continually create new models, the models they creates must be usable for a period of time. This means that the usage of the texts must be fairly stable.

In summary, asking domain experts to build models that approximate how a group of users will use texts can work if the texts are used in ways that are predictable, limited, and stable; and if the user model is understandable to the domain expert. We expect this may be true in other NLG applications in addition to weather forecasts (financial summaries?), and encourage other researchers to consider this approach when it seems appropriate.

5 GIRL: Obtain Model by Testing Users

The final approach we have tried is building a model of a user's skills by testing his performance on a set of tasks, using an independently developed assessment test.

We are using this approach in a new system, GIRL (Williams, 2002), which generates reports on how well a student has done in a computer-based literacy assessment. From a research perspective, GIRL's focus is on making microplanning choices (aggregation, word choice, etc.) that are appropriate for the recipient's reading ability. For example, an aggregation decision that leads to a 30-word sentence is acceptable for a good reader but not for a poor reader. This requires knowing how well the recipient can read, and GIRL obtains this information from the literacy assessment which the student has completed. The assessment was independently developed by NFER-Nelson as a skill-based reading, writing and listening test for adult literacy learners. The results give a hierarchical model of derived literacy skills, with results of individual questions at the leaf nodes, results of skills tests at the next level, overall reading, overall writing and overall listening levels at the next and the overall literacy level at the root of the tree. Part of the experimental work in GIRL is determining which information from this model is most useful in guiding expression choices. The ultimate vision in GIRL is to create user models for readers which include this key information, and then use the reader model to control microplanning and perhaps also content and realisation choices in a text-generation system.

We cannot say much about the success or failure of this approach yet, as GIRL is still being developed and has not yet been evaluated. But certainly it seems likely that we cannot expect to obtain a large amount of information using this strategy; like the STOP strategy of asking a user to fill out a questionnaire, the strategy of testing the user is feasible if a small amount of information is needed, but not otherwise.

It is possible that in the long term, a lot of observational data about users will be available, which perhaps can be analysed and 'mined' for user information (Fitzgibbon and Reiter, 2002). For example, if we knew what web pages a person had read, we could probably make a plausible guess about his or her literacy level. This is an interesting possibility which should be kept in mind for the future, even if it is not realistic now.

6 Discussion

6.1 Understandability of User Models

One recurring theme in our work is that user models should be understandable to users or domain experts. If we had perfect user models, users perhaps would not care how they worked; but since we have imperfect user models, it is very helpful if users can understand them, either to edit them or to understand what the system is doing and how it might fail.

Kay (2001, page 118) discusses giving users access to and control over models about themselves in intelligent tutoring systems, and argues that this

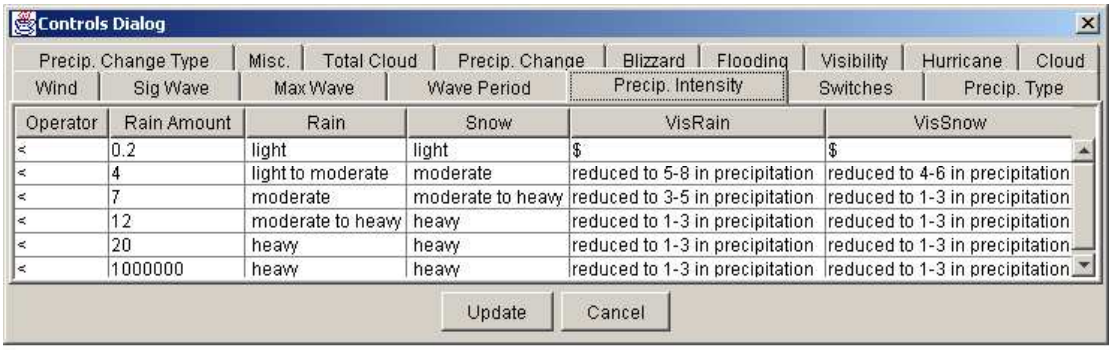

Figure 1: Precipitation Intensity Options Box from SUMTIME-MOUSAM

is desirable because:

- Users have a legal and ethical right to access and control about themselves.
- Users can assess the correctness of the model, and repair mistakes.
- Users who have access to their user models will have a better idea of what the system is doing.
- Developers will be more accountable if models can be inspected by users, and will place more importance on making models understandable.
- Users may learn useful information about themselves from their user models.

Although the last reason may be specific to tutoring systems, we believe that the first four rationales give by Kay for giving users control over their models are likely to apply to NLG systems that use user models as well.

Many commercial software packages allow users to specify their preferences and needs via an options box, and we have in fact done this to a limited extent in SUMTIME-MOUSAM. A simple SUMTIME-MOUSAM options box is shown in Figure 1; this box allows the domain expert to specify how numerical precipitation figures (in mm/hr) should be translated into linguistic descriptors such as *heavy*. This is user-dependent because different readers interpret *heavy* in different ways (Reiter and Sripada, 2002). Among other things, the interpretation depends on location; a precipitation amount that would be considered *heavy* in a dry environment such as the Middle East might not be considered *heavy* in a wet environment such as Scotland.

Perhaps the key technical challenge to building such options boxes or other model-editing facilities is making choices understandable to users and domain experts. In particular we cannot expect such people to have expertise in linguistics; this is why the descriptors in Figure 1 are specified as actual adjectival phrases instead of as conceptual or semantic representations. We also cannot expect such people to have expertise in computer science; this is an issue in some of the other SUMTIME-MOUSAM options boxes, where the expert is expected to specify parameters that control a linear segmentation algorithm (Sripada et al., 2002). Finally, if the options box is intended to be filled out by users instead of domain experts, we should not expect an unrealistic amount of domain knowledge. For example the options box in Figure 1 may not be understandable to end users, because they may not understand precipitation expressed as mm/hr.

In short, we need to be able to present user model choices in an understandable way to people who do not have expertise in linguistics or computer science; how best to do this is an important topic for future research. This problem is likely to become even more acute if we try to learn user models from large amounts of observational data, as many learning algorithms do not produce results which are easy for people to understand.

6.2 Knowledge Source: Users or Domain (Communication) Experts

An important related issue is who supplies the information for a user model. In IDAS, STOP and GIRL the users themselves supply the basic information (which the system may make inferences from), but in SUMTIME-MOUSAM the information comes from a forecaster, that is a domain expert who knows the users. The forecaster is also a *domain communication* expert (Kittredge et al., 1991); that is, he is an expert in how to communicate information about the weather to users, as well as an expert in meteorology itself.

The disadvantage of using a domain (communication) expert is that he is supplying information secondhand, he does not know the users as well as the users know themselves. If we consider Figure 1, for example, presumably the users themselves have a better idea of what *heavy* means to them than the domain expert does. But on the other hand, the advantage of using a domain expert is that he will have more domain knowledge (for example, understand precipitation expressed as mm/hr), more understanding of how the NLG system works and how it is controlled by the user model, and also more domain communication knowledge (for example, know how *heavy* was defined for other users). Because he understands the domain, the user, and the system, the domain expert can be a good person to create a model that 'bridges the gap' between the user and the system, and specifies to the system, in terms it can understand, what is important to the user and how best to communicate with the user linguistically.

The other reason for acquiring user models from domain experts in SUMTIME-MOUSAM was that SUMTIME-MOUSAM weather forecasts are read by many people in an oil rig, not just a single individual. Hence the user model needs to be an approximation to a group of people, not a detailed description of a single person. In other words, the user model should record how the recipients on average interpret *heavy*, not how one individual interprets this word. In SUMTIME-MOUSAM we believed that it might in fact be easiest for an outside expert to create such a model, especially if the expert had experience in doing this for other user groups.

6.3 Cost of Mistakes

Another general issue that needs to be considered is the impact of getting the user model wrong. For example, what is the impact of careless mistakes in STOP questionnaires, of users failing to give the SUMTIME-MOUSAM domain expert complete information about all of their tasks and lexical interpretations, or of a GIRL subject doing poorly on a test because she hadn't had much sleep the previous night? Do such mistakes marginally decrease the utility of generated texts, or do they make generated texts completely useless? NLG systems that are not robust in the face of such mistakes may not be useful in real-world applications.

This may be an especially serious problem for systems such as STOP and GIRL which make inferences about the user based on the information he or she explicitly specifies. For example, if a smoker does not realise that STOP uses the 'do you smoke within 30 minutes of waking up' question to infer addiction level, and instead believes that this question is unimportant, then he or she may ignore it or respond very quickly without thinking about the question.

This suggests that at minimum we need to carefully design the user model acquisition tool (STOP questionnaire, SUMTIME-MOUSAM option boxes, GIRL literacy test) to minimise careless errors. It may also be sensible to check user models for plausibility and consistency (as STOP in fact does). Finally, this once again emphasises the desirability of users understanding what is in the user model and how the system uses it.

A related issue is how to update and maintain user models. People of course change over time, and user models should change with them. We are not aware of any previous research on how user models for NLG systems should be updated to reflect changes in the user's expertise or preferences. Probably the simplest approach here is simply to let users or domain experts directly edit and update a user model, which is possible in SUMTIME-MOUSAM.

7 Conclusion

One of the great promises of NLG is that it can tailor texts for individual users, but realising

this promise requires that we know a lot about users, and in practice it can be difficult to obtain detailed information about the expertise, background, tasks, goals, and so forth of individual users. We have tried various approaches to acquiring information about users for NLG systems, including letting users implicitly specify models (IDAS), explicitly asking users to enter a model (STOP), explicitly asking a domain expert to construct a model (SUMTIME-MOUSAM), and implicitly inferring a model from a standard assessment test (GIRL). None of these approaches seem generally applicable, but all probably work better if users can easily understand their models, so that they can edit their model or at least better understand what the NLG system is doing.

Acknowledgements

Many thanks to Chris Mellish, Ingrid Zukerman and the anonymous reviewers for their helpful comments. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC), under a PhD Studentship and grants GR/F36750/01, GR/L48812 and GR/M76681; and by the Scottish Office Department of Health under grant K/OPR/2/2/D318.

References

Doug Appelt. 1985. *Planning English Referring Expressions*. Cambridge University Press, New York.

John Bateman. 1997. Enabling technology for multilingual natural language generation: the KPML development environment. *Natural Language Engineering*, 3:15–55.

Gregg Collins and Louise Pryor. 1995. Planning under uncertainty: Some key issues. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1567–1573.

Michael Elhadad and Jacques Robin. 1997. SURGE: A comprehensive plug-in syntactic realisation component for text generation. Technical report, Computer Science Dept, Ben-Gurion University, Beer Sheva, Israel.

Gerhard Fischer. 2001. User modelling in human-computer interaction. *User Modeling and User-Adapted Interaction*, 11:65–86.

Andrew Fitzgibbon and Ehud Reiter. 2002. Memories for life: Managing information over a human lifetime. Technical Report AUCS/TR0207, Department of Computing Science, University of Aberdeen, UK.

Peter Haddawy, AnHai Doan, and Richard Goodwin. 1995. Efficient decision-theoretic planning: Techniques and empirical analysis. In *Proc. Eleventh Conf. on Uncertainty in Artificial Intelligence*, pages 229–236.

Todd Heatherton, Lynn Kozlowski, Richard Frecker, and Karl-Olav Fagerström. 1991. The Fagerström test for nicotine dependence: A revision of the Fagerström tolerance questionnaire. *British Journal of Addiction*, 86:1119–1127.

Judy Kay. 2001. Learner control. *User Modeling and User-Adapted Interaction*, 11:111–127.

Richard Kittredge, Tanya Korelsky, and Owen Rambow. 1991. On the need for domain communication language. *Computational Intelligence*, 7(4):305–314.

John Levine and Chris Mellish. 1995. The IDAS user trials: Quantitative evaluation of an applied natural language generation system. In *Proceedings of the Fifth European Workshop on Natural Language Generation*, pages 75–93, Leiden, The Netherlands.

Johanna Moore and Cecile Paris. 1993. Planning text for advisory dialogues. *Computational Linguistics*, 19:651–694.

Ehud Reiter and Somayajulu Sripada. 2002. Human variation and lexical choice. *Computational Linguistics*, 28:545–553.

Ehud Reiter, Chris Mellish, and John Levine. 1995. Automatic generation of technical documentation. *Applied Artificial Intelligence*, 9(3):259–287.

Ehud Reiter, Roma Robertson, and Liesl Osman. 2003. Lessons from a failure: Generating tailored smoking cessation letters. *Artificial Intelligence*. in press.

Somayajulu Sripada, Ehud Reiter, Jim Hunter, and Jin Yu. 2002. Segmenting time series for weather forecasting. In *Applications and Innovations in Intelligent Systems X*, pages 105–118. Springer-Verlag.

Sandra Williams. 2002. Natural language generation of discourse connectives for different reading levels. In *Proceedings of the 5th Annual CLUK Research Colloquium*.

Ingrid Zukerman and Diane Litman. 2001. Natural language processing and user modeling: Synergies and limitations. *User Modeling and User-Adapted Interaction*, 11:129–158.