Generation of Adaptive (Hyper)Text Explanations with an Agent Model

Kalina Bontcheva, Yorick Wilks
Department of Computer Science, University of Sheffield
Regent Court, 211 Portobello Street, Sheffield S1 4DP, UK
{kalina,yorick}@dcs.shef.ac.uk

Abstract

A number of generation systems have chosen hypertext as a target media and study ways of adapting its content and links to the presentation context and the particular user. Our experience with hypertext generation has shown that a user model with beliefs, goals and interests could help to improve the explanation quality. Therefore, an existing agent modelling framework is being integrated within a hypertext generation system (HYLITE). This paper discusses its use and constraints on the generation process.

1 Introduction

With the growing information content and popularity of the World-Wide Web came the need for automatic construction of user-sensitive hypertext (e.g. personalised space on Web portals). Since these issues also fall in the domain of Natural Language Generation (NLG), several systems (e.g., ILEX [8], PEBA II [13]) started exploring the applicability of NLG techniques to *dynamic hypertext* generation[1]. Since users expect real-time interaction, efficient and robust generation techniques are often applied. For instance, ILEX uses a combination of canned stories and templates; EXEMPLARS [19] is rule-based; and PEBA II uses text schemas [12] and a phrasal lexicon. Their User Models (UM) mostly represent different classes of users (e.g., expert, novice in PEBA II) which allows for some variation of hypertext content and links.

However, our experience with hypertext generation (section 2) has shown that more detailed user information, a kind of user profile with beliefs, goals and interests, might allow better adaptivity. Also, since several users can interact simultaneously and independently with the system, multiple profiles need to be maintained, stored and accessed simultaneously. Therefore, we are now studying the use of a generic agent modelling framework for adapting generated explanations. Such frameworks provide the required functionality and expressive power and also offer improved scalability and re-use.

To this end, we will take an existing belief modelling framework – ViewGen (section 3) – and discuss its use within our language generation system. Some advantages of this particular framework are its orientation towards representation and maintenance of attitudes held by dialogue participants; and its previous use for metaphor and speech act understanding [1, 10].

The mechanism for acquisition and dynamic update of user beliefs is presented in section 4. The hypertext generation process is discussed in section 5 together with a brief example (section 6). Section 7 places this approach in the context of previous work on user modelling for language generation and section 8 draws some conclusions and outlines future work.

[1] In *dynamic hypertext* node content and links are created on demand and are often adapted to the user's interests and previous interaction.

2 The Baseline System

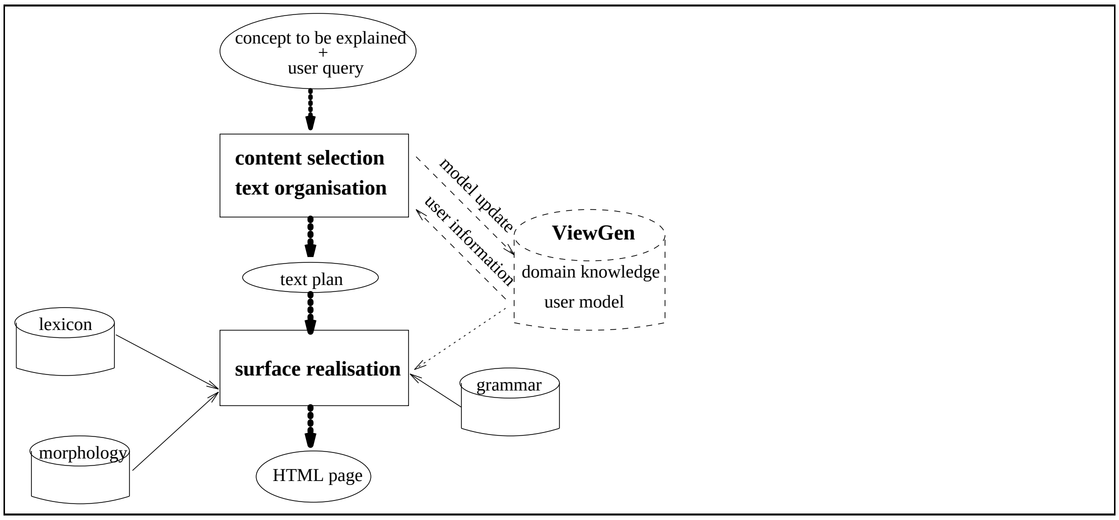

Figure 1: System Architecture

2.1 System Architecture

We started by building a baseline hypertext generation system – HYLITE – that does not have user information and a record of previous interaction. It generates brief encyclopedic hypertext explanations (see figure 2) in response to user requests for information. The NLG system runs as an independent process on the server and the user interacts with it via an Internet browser and a CGI script.

An important characteristic of HYLITE is that it is aimed at providing encyclopedic-style information in a flexible and adaptive manner but it is not an intelligent tutoring environment. It also facilitates the browsing user in finding the necessary factual information and navigating around the hypertext space, which in effect, means navigating through the underlying domain knowledge.

The architecture of the generator is shown in figure 1. Following Reiter [17], it can be classified as a pipeline model, with separate *content planning* and *surface realisation* modules.

The text planner uses high-level discourse patterns similar to text schemas [12] which have been derived from analysing entries in encyclopaedia and terminological dictionaries. For instance, entities (i.e., concepts inheriting from `ENTITY` in the hierarchy) are defined by their *supertype(s) or type definition, characteristics, functions, constituents* and *examples*. If the entity has several synonymous terms, the query one is used throughout the explanation and the rest are given in brackets when the entity is first introduced.

Even though this approach has some known limitations (e.g., see [15]), it proved sufficient for generating short terminological explanations and particularly useful due to its computational efficiency. In addition, the user-controlled, information-seeking interaction with HYLITE does not require the sophisticated feedback and recovery mechanisms necessary for advise-giving or intelligent tutoring systems (e.g. [15, 5]).

The current implementation runs on a knowledge base in a chemical sub-domain. A bigger experiment in a computer science sub-domain is under development. We are using an existing German surface realiser, modified for English [3]. The generated explanations are paragraph-sized and contain hypertext links to related concepts (see figure 2 for a short example).

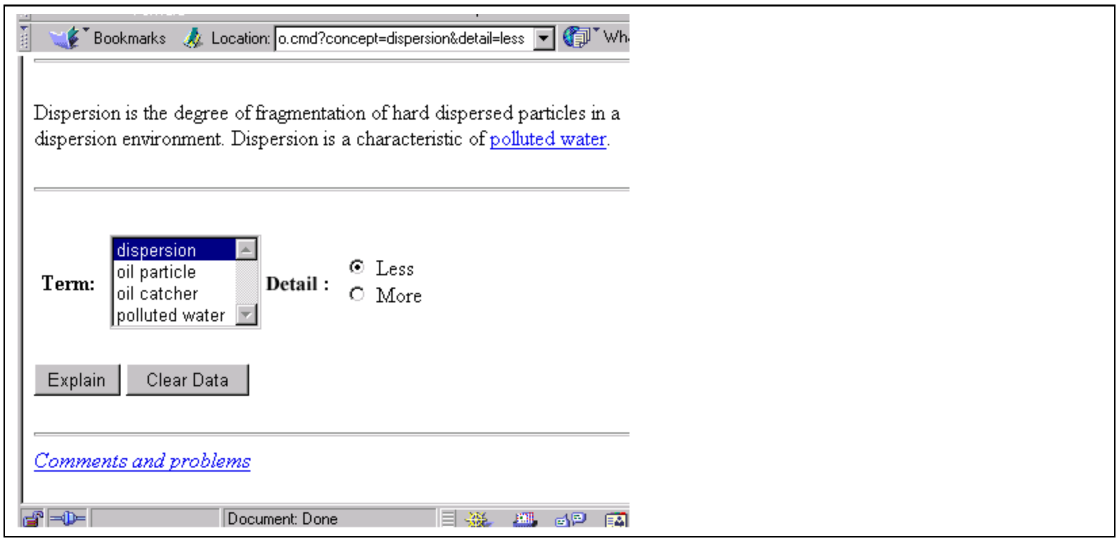

Figure 2: A short explanation of *dispersion*

2.2 Problems with the Explanations

As evident from the example, since the generator has no information about user's knowledge, it cannot avoid defining an unknown concept – dispersion – using other unknown terms – e.g. dispersion environment. In addition, if we take this example one step further and follow the **`polluted water`** link, the system generates the following explanation:

> **`Polluted water is water. Polluted water is characterised by dispersion, concentration and resistance...`**

The first problem is including trivial information in the first sentence. Also, the explanation would sound more natural if the system has taken into account that the user has already been told about dispersion. In such circumstances, a sentence like *Apart from dispersion, concentration and resistance are also characteristics of polluted water* would be more appropriate because it links known information with the new one.

Therefore, in order to improve the quality of the explanations, the system needs a user model that tracks dynamically known facts and concepts and distinguishes between common and specialised knowledge. It also needs to store multiple, independent user profiles since several users might interact with the system simultaneously over the Web. Further analysis of the generated hypertext also revealed problems with the simple novice-expert distinction so a more flexible stereotype mechanism would be advantageous.

However, as already experienced by others, e.g. Kobsa [9], building a sophisticated user modelling component to address specific application needs can be very time-consuming. Reusability and scalability also motivated research and use of generic, application-independent user modelling frameworks, also called *user modelling shells*. Such shells provide mechanisms for storage, retrieval and maintenance of user-related information. The application is only responsible for populating, updating and quering the shell which effectively acts like a user-modelling "black-box".

Consequently, HYLITE has been extended to include such a generic agent[2] modelling framework – ViewGen – which is used for adaptive content selection and text organisation. Potentially, it can be also used during surface realisation (e.g., lexical choice) but this falls outside the scope of this work: hence, the dotted line connection in figure 1.

[2] ViewGen is an *agent* not a *user* modelling framework because it can model the (nested) attitudes of *many* dialogue/discourse participants (i.e., agents) as opposed to just one user.

The impact of the agent modelling framework on the system performance and explanation quality will be evaluated in comparison to the baseline system. In fact, such incremental development prevents the system from being dependent on the availability of user-related information. This needs to be avoided, since as argued by Sparck-Jones [18], adaptivity is useful but NLG systems should be able to produce acceptable texts even without such a model.

3 Agent Modelling with ViewGen

ViewGen [1, 10] is an agent modelling framework which represents beliefs,[3] intentions and goals of dialogue/discourse participants. It represents such attitudes in *nested environments* (see figure 3). There are two types of environments – *viewpoints*, which represent someone's beliefs, goals or intentions (e.g., John's beliefs) and *topic* environments, which contain propositions about a topic (e.g., facts about New Mexico)[4]. Propositions can appear in more than one environment.

In HYLITE+, all topic environments correspond to domain concepts and there are no nested agent viewpoints (since users only interact with the system and are not aware of each other). Therefore, ViewGen is mostly concerned with constructing and maintaining independent user viewpoints and the topic environments they contain. As there is no inter-agent interaction, many intentionality issues do not arise (e.g., whether Frank's father is John for all agents) and the ViewGen environments are simpler. Nevertheless, more sophisticated ones can be added if necessary.

From an NLG perspective, topic environments provide a mechanism for restricting the search space during content selection. Effectively, given a concept, ViewGen identifies all relevant environments and then searches only *within* them (see section 5).

For efficiency reasons, topic environments and viewpoints are built on demand by a process of *ascription*. Ascription assumes that attitudes held by one agent can be ascribed to others. There are two main methods of ascription – *default* ascription and *stereotypical* ascription [2]. Default ascription applies to common attitudes which ViewGen assumes that any agent will hold and also ascribe to any other agent *unless there is contrary evidence* (see figure 3). Stereotypical ascription applies to 'uncommon' attitudes which ViewGen assumes hold only for agents from a particular stereotype.

Additionally, relevant stereotypes are not applied just once in the beginning of the interaction; instead, they are *consulted* with respect to a specific belief or topic only when necessary. This also provides flexibility in case of conflicting information, i.e., one stereotype could be preferred for a given topic, while another stereotype might be preferred for the next.

ViewGen encodes all domain knowledge within the main system environment. The type hierarchy contains both commonly known concepts (the upper part) and domain specific ones. Common concepts are subject to default ascription, e.g., ViewGen will assume that every agent knows that tables are physical objects. Domain-specific concepts appear only within stereotypes and can be ascribed to an agent only if that stereotype is applicable. For instance, ViewGen will ascribe the proposition that an emulsion is a colloid only to agents believed to know about chemistry.

Both ascription methods are examples of default reasoning and some mechanism is required for dealing with conflicting ascriptions and belief revision. To handle such problems, ViewGen uses a truth maintenance system (TMS) [7].

4 Acquiring and Updating User Beliefs

HYLITE+ has an initial login screen where the user identifies herself. Based on that, her ViewGen environment is created or located if already exists. Each new user is also presented with the option to provide some data about herself e.g. occupation, topics of interest. This

[3] Beliefs are the propositions held by an agent to be true.

[4] For a description of the representation of goals and intentions, as well as the planning mechanism see [10]

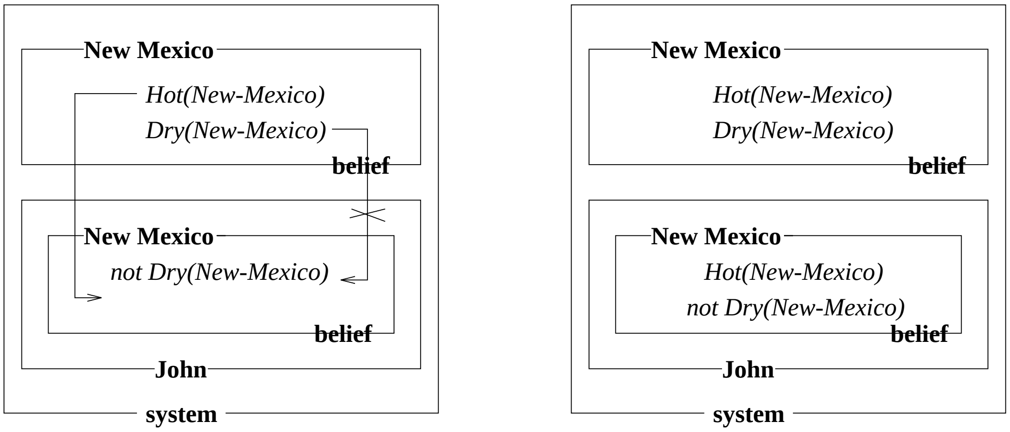

Figure 3: Examples of ViewGen belief environments: before ascription (left) and after ascription (right)

information, if available, can then be used to determine which stereotypes might apply. When no information is given, no such assumptions can be made about the user so she is considered to have only common-sense knowledge that can be ascribed by default to any agent.

As the interaction progresses, user beliefs are updated in response to system and user behaviour, thus recording the *interaction history*.

After each explanation HYLITE+ provides ViewGen with information about which domain facts under which topic have been included in the generated explanation. In this way ViewGen can update dynamically the user environment by ascribing relevant beliefs.

For the `dispersion` explanation example (figure 2), partial states of the ViewGen environments are shown in figure 4. The left-hand side shows that initially the user (John) has no beliefs about `dispersion` and none can be ascribed (hence the crossed arrow) as this is not common knowledge and no relevant stereotypes apply. Therefore HYLITE+ selects relevant facts (see section 5) from the system's `dispersion` environment (obtained from ViewGen) and generates an explanation. It also informs ViewGen to update the user beliefs about dispersion (see figure 4 – right).

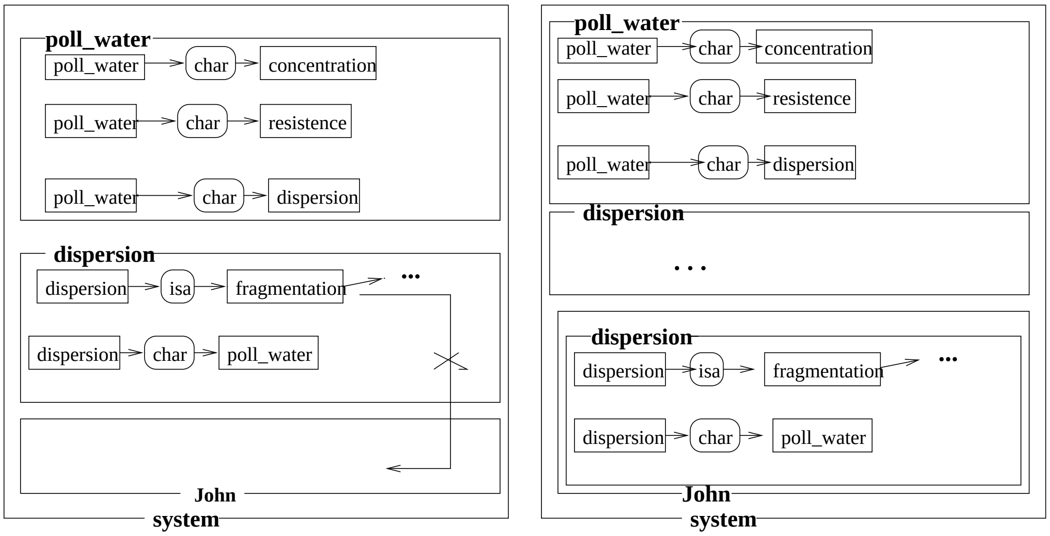

Figure 4: A user's environment before (left) and after an explanation about dispersion (right)

This example highlights several points. First, in HYLITE+ user interaction and feedback are carried out through hypertext browsing and searching, in an information-seeking context. Therefore the system can only collect user-related information in an unobtrusive manner, by monitoring and reasoning about user's actions and explanation requests. This somewhat limits the assumptions about user beliefs which can be made by ViewGen based on the information passed from the generator.

Another aspect is the way HYLITE+ interprets user beliefs obtained from ViewGen environments. Since some beliefs have been ascribed following an explanation, the generator should take that into account as the user is already familiar with them. However the system does not expect or rely that the user indeed has such beliefs as would be important in an intelligent tutoring context. Nevertheless, HYLITE+ needs to distinguish such beliefs from beliefs which have been ascribed from a stereotype or inferred from other beliefs by ViewGen. This distinction is necessary for subsequent explanations (see the polluted water example), where HYLITE+ can refer to already mentioned facts and also know whether the user is returning to a previously explained, as opposed to supposedly known, concept.

5 Content Selection using ViewGen

The user requests explanations by selecting unknown concepts and questions (e.g. 'What is?'). The system initiates a process of selecting an explanation content by taking into account the request and the user:

1. **Extract relevant propositions** from the system's viewpoint given the user request, i.e., a domain concept or an instance of such a concept.

2. **Determine** which are **already known** by the user and mark them as such to enable the text planner and surface realiser to take that into account (see section 6). In case of contradictory beliefs between system and user (see New Mexico example), both system and user beliefs are added to the propositions to be conveyed (e.g. *believe(John,not-Dry(New Mexico))*), so the generator might use/contrast the contradictory beliefs in the explanation.

3. **Inspect each proposition** for unknown related concepts and, **iteratively**, retrieve relevant information for them if not constrained by time or space. Mark concepts as unknown otherwise[5].

5.1 Extracting relevant propositions

Given a request for explanation of a domain concept C, the system has to obtain a set of propositions which are relevant and also unknown to the user. Therefore, ViewGen has to determine first which topic environments from the system main viewpoint are relevant and need to be searched.

Searching for relevant environments: For efficiency reasons, the search process is based on inheritance and restricted to some meaningful relations, e.g., part-of, instrument, agent. Relevant environments are considered to be:

1. Topic environments with the concept C contained in their topics and environments with topics more generic than the concept C – determined on the basis of the type hierarchy. For example, if C is `Java`, relevant environments would be *Java* and *programming_language*.

2. If previous search fails, search all topic environments for propositions where C appears in `CHAR`, `AGNT`, `PTNT`, `INST` and `PART-OF` relations. If successful, construct and return an environment with C as a topic and all such propositions as content.

[5] At a later stage, some marked concepts are realised as hypertext links which the user can follow for further explanations.

Deciding on relevant propositions: Once the relevant topic environments have been identified, all propositions contained in them are deemed potentially relevant. However, some 'normalisation' of the propositions might be required:

- removing repetitions and more generic propositions subsumed by more specific ones;
- replacing occurrences of supertypes of the query concept C with C in propositions inherited from a more generic topic.

5.2 Accounting for user beliefs

Given the set of relevant propositions from the previous stage, the generator needs to account for all propositions believed to be familiar to the user. Therefore each proposition is looked up in ViewGen and marked in the set as already believed by the user.

5.3 Including additional information

Finally, the system inspects each proposition and checks with ViewGen the user beliefs for each 'important' concept linked to the query concept C (e.g, concepts linked with agent, part-of, or instrument relations). ViewGen considers a concept unknown if the user viewpoint contains no beliefs about it and none can be ascribed from a stereotype or by default. In such cases, the generator marks these concepts and, if possible, iteratively obtains relevant propositions from ViewGen and adds them to the current set.

6 An Interaction Example

Let us consider that initially the user (John) has requested a definition of `dispersion` and the system provided the explanation in figure 2. The ViewGen environments before and after the explanation are shown in figure 4. If afterwards the user follows the `polluted water` link, HYLITE+ needs to define polluted water. Let us suppose that system beliefs about polluted water are these contained in the respective environment (figure 4).

The generation system has to decide what is the relevant content, so it asks ViewGen about John's beliefs about polluted water. ViewGen responds that John does not know anything about polluted water, since John's environment and all applicable stereotypes do not contain such information. Then HYLITE+ decides that the content of the system's **poll_water** environment should be conveyed.

Afterwards, by examining the facts in this environment, e.g., the fact about polluted water being characterised by dispersion, it tries to establish whether the user is familiar with the concept. This time ViewGen's model of user beliefs contains such environment and therefore it replies that dispersion is familiar. HYLITE+ then checks with ViewGen and establishes that the whole fact is already familiar to the user from a previous explanation and marks it as such.

In the same way the system checks whether the user is familiar with concentration and resistance, which she is not. In general, HYLITE+ can consult ViewGen for each relevant concept it considers important.

Then if the system is in verbose mode, e.g., the user has requested **more** detail (see figure 2) it will extract recursively all relevant facts about resistance and concentration. When **less** detail is selected, the concepts are just marked as unknown and hypertext links to their explanations might be provided.

In this case, as a result of the content selection process, the system determines the following three facts to be used for the explanation:

[POLL_WATER]->(CHAR)->[DISPERSION] : **known**
[POLL_WATER]->(CHAR)->[RESISTENCE] : **unknown**
[POLL_WATER]->(CHAR)->[CONCENTRATION] : **unknown**

To underline the repetition and make a reference to the previous explanation, the content can be realised as:

> *Apart from dispersion, other characteristics of polluted water are concentration and resistance.*

In other circumstances, the known fact(s) may not be included in the explanation in order to avoid repetitions. However, in this case, the generator prefers to include the fact but mark it in the surface form by using an *Apart of...* construction. This decision is based on the intuition that it is preferable to include exhaustively all characteristics, parts and functions of a concept in order to avoid misleading the user.

7 Discussion

Even though we argued for using a generic agent modelling framework for generation of adaptive **hypertext**, this approach is by no means restricted to hypertext as a media. Previous work on text generation has often used and sometimes maintained a user model in order to provide tailored output – e.g. TAILOR [16], ROMPER [11], PEA [14]. Most systems tend to use an *overlay* model where facts from the generator's knowledge base can be marked as known by the user. Differences and conflicts in user and system beliefs are not represented with a few exceptions, e.g., ROMPER. Often the representational power and structure of the user model are chosen to be sufficient for the task at hand. However, none of these systems have attempted to use an independent user modelling framework which offers lower implementation overhead and better portability and re-use. Therefore, this work is also relevant to text generation as it provides an insight into the viability of this approach.

8 Conclusion

In this paper we presented a hypertext generation system which uses ViewGen – a framework for modelling the attitudes of dialogue participants – as one of its knowledge sources. The impact of the component on content selection has been discussed. HYLITE+ is currently under evaluation against the baseline system in an attempt to measure the costs and benefits of using an agent modelling framework. As argued earlier, the use of efficient and robust generation techniques was necessary due to the real-time interaction with the users. The main difference from other generation approaches is the use of a generic agent modelling framework instead of a custom-made user model.

Previous attempts at *evaluating* language generation systems [6] have shown that it is a difficult task and that the results seldom measure the performance and contribution of *each* module within the system. A measure of the overall performance can be obtained by using a black-box evaluation assessing the 'quality' of the generated hypertext based on users feedback. Since we are mostly interested in the role of the modelling framework, we intend to run the generation system on the same domain knowledge both *with* and *without* ViewGen and then compare execution time and output quality.

We are also working on problems specific to hypertext as a presentation media [4]: what should be a link; how many links to have on a page; when to explain an unknown concept in the text and when to prefer a link; how to present previously seen material.

In the future, we would like to experiment with generating explanations on the same topic from different perspectives. Also, as already discussed, in the current information-browsing settings, agents are not aware of each other. Therefore, another line of research would be to use ViewGen for generation in a multi-agent communication environment, e.g., dialogues. A dialogue system would also allow us to study the shared use of ViewGen by two tasks — understanding and generation.

Acknowledgements

We wish to thank Hamish Cunningham, Robert Gaizauskas and the anonymous referees for the helpful comments on this work.

References

[1] Afzal Ballim and Yorick Wilks. *Artificial Believers.* Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1991.

[2] Afzal Ballim and Yorick Wilks. Beliefs, stereotypes and dynamic agent modelling. *User Modelling and User-Adapted Interaction*, 1:33 – 65, 1991.

[3] Kalina Bontcheva. Generation of Multilingual Explanations from Conceptual Graphs. In Ruslan Mitkov and Nicolas Nicolov, editors, *Recent Advances in Natural Language Processing: Selected Papers from RANLP'95*, volume 136 of *Current Issues in Linguistic Theory (CILT)*. John Benjamins, Amsterdam/Philadelphia, 1997.

[4] Kalina Bontcheva and Yorick Wilks. Combining language generation and belief modelling into a flexible hypertext system. In *Proceedings of the Flexible Hypertext Workshop*, Southampton, UK, April 1997. A workshop held in conjunction with the Eight ACM International Hypertext Conference (Hypertext'97).

[5] Alison Cawsey. *Explanation and interaction: the computer generation of explanatory dialogues.* Cambridge, Mass.; London:MIT Press, 1992.

[6] Robert Dale and Chris Mellish. Towards evaluation in natural language generation. In *Proceedings of First International Conference on Language Resources and Evaluation*, Granada, Spain, 28-30 May, 1998.

[7] John Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231 – 272, 1979.

[8] Alistair Knott, Chris Mellish, Jon Oberlander, and Mick O'Donnell. Sources of flexibility in dynamic hypertext generation. In *Proceedings of the 8th International Workshop on Natural Language Generation (INLG'96)*, 1996.

[9] Alfred Kobsa, Andreas Nill, and J. Fink. Hypertext and hypermedia clients of the user modelling system BGP-MS. In Mark Maybury, editor, *Intelligent Multimedia Information Retrieval.* MIT Press, 1997. To appear.

[10] Mark Lee and Yorick Wilks. An asciption-based approach to Speech Acts. In *Proceedings of the 16th Conference on Computational Linguistics (COLING-96)*, Copenhagen, Denmark, 1996.

[11] Kathleen F. McCoy. Reasoning on a dynamically highlighted user model to respond to misconceptions. *Computational Linguistics*, 14 (3), September 1988.

[12] Kathleen R McKeown. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press, Cambridge, England, 1985.

[13] Maria Milosavljevic, Adrian Tulloch, and Robert Dale. Text Generation in a Dynamic Hypertext Environment. In *Proc. of 19th Australian Computer Science Conference*, Melbourne, 1996.

[14] Johanna D. Moore. *Participating in Explanatory Dialogues.* MIT Press, Cambridge, MA, 1995.

[15] Johanna D. Moore and Cécile L. Paris. Planning texts for advisory dialogs: capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651 – 694, December 1993.

[16] Cécile L. Paris. *User modelling in text generation.* Francis Pinter Publishers, London, 1993.

[17] Ehud Reiter. Has a Consensus NL Generation Architecture Appeared, and is it Psycholinguistically Plausible? In *Proceedings of 7th Int. Workshop on NL Generation (INLG-94)*, pages 163–170, Kennebunkport, Maine, USA, 1994.

[18] Karen Sparck Jones. Tailoring output to the user: What does user modelling in generation mean? In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 201–226. Kluwer Academic Publishers, Boston, 1991.

[19] Michael White. Designing dynamic hypertext. In *2nd Workshop on Adaptive Hypertext and Hypermedia*, June 1998. Held in conjunction with Hypertext'98, Pittsburgh, USA.