Generating Tutorial Feedback with Affect

Johanna D. Moore[*], Kaska Porayska-Pomsta[*], Sebastian Varges[‡], and Claus Zinn[*]

[*]University of Edinburgh
School of Informatics
2 Buccleuch Place
Edinburgh, EH8 9LW, UK
{jmoore,kaska,zinn}@inf.ed.ac.uk

[‡]University of Brighton
Information Technology Research Institute
Lewes Road
Brighton, BN2 4GJ, UK.
Sebastian.Varges@itri.brighton.ac.uk

Abstract

Studies aimed at understanding what makes human tutoring effective have noted that the type of indirect guidance that characterizes human tutorial dialogue is a key factor. In this paper, we describe an approach that brings together sociolingusitic research on the basis of linguistic choice with natural language generation technology to systematically produce tutorial feedback appropriate to the given situation.

Introduction

One-on-one human tutoring has been shown to be the most effective type of learning intervention (Bloom 1994). Debates about what makes human tutoring effective, and how this could be captured in computer-based learning environments have led to detailed studies of human tutoring. The consensus from these studies is that experienced human tutors maintain a delicate balance, allowing students to do as much of the work as possible, while providing them with enough guidance to keep them from becoming frustrated or confused (Fox 1993, Lepper & Chabay 1988). For example, Fox found that tutors provide frequent feedback indicating that students' problem solving steps are okay. A short hesitation in responding "okay'" typically led the student to assume that something was amiss with the current step, and frequently led students to repair their own errors. When more explicit help was required, the tutor focused the student's attention on the part of their solution that required modification or on information that would be useful in repairing the error. In general, tutors try to avoid telling the student that they are wrong or precisely how a step is incorrect. Instead they try to lead students to discover the error and to repair it themselves. This type of indirect guidance allows students to maintain a feeling of control, and there is evidence that it has strong motivational benefits.

Although researchers studying tutorial dialogues have devised inventories of the types of feedback strategies that human tutors employ, there is no systematic account of the linguistic forms that are used to realize feedback strategies (either positive or corrective), and there is little understanding of what factors influence linguistic choice. In this paper, we describe how we have adapted work from sociolinguistics to define strategies for linguistic choice based on a model of the student and the problem solving situation. We describe how we have operationalized these strategies in an implemented intelligent tutoring system.

Theoretical basis of linguistic choice

Given the goal of achieving a particular communicative intention in a specific social setting, an agent must choose among all the possible variations in semantic content, syntactic form and acoustic realization. Brown and Levinson (1987), henceforth B&L, provide a theory of social interaction in which they identify variables that affect choice, and specify how different values for the variables produce different communicative outcomes. In their theory, all members of society have (and know each other to have):

(1) Face: public self-image, consisting of:
   a. Negative face: A need for freedom of action and freedom from imposition, i.e., a desire for *autonomy*, and
   b. Positive face: A positive self-image that is appreciated and approved of by others, i.e., a desire for *approval,*
(2) Capabilities for rational reasoning, in particular means-end reasoning.

According to B&L, face regulates speakers' linguistic choices at all times. People typically choose their utterances to minimize threat to their own and others' face. In B&L's model threat is calculated as a sum of three socially determined variables: (1) the social distance between the speaker and the hearer, (2) the power that the hearer has over the speaker, and (3) a ranking of imposition for the act under discussion. Humans use personal experience, cultural norms and situational factors to determine the values for these variables at any given time. For example, social distance depends on things like how well S and H know one another, or whether they belong to the same peer group. Power is determined by a variety of sources, such as status or the ability of S to control access to goods that H wants (e.g., information or money). Acts (e.g., requests, commands) are ranked according to how

much they interfere with an agent's desires for self-determination and approval. A speaker's ability to assess the situation with respect to a hearer's social, cultural and emotional needs and hence his ability to produce polite language (i.e., to engage in *facework*) constitutes a crucial facet of his social and linguistic competence.

Specific values of the social variables are encapsulated in the socio-cultural conventions (politeness rules) of a speech community in which a given language is produced. The conventions are manifested in concrete communicative strategies that people in a given speech community ought to use when attempting to produce socially and culturally acceptable language (Fetzer, 2003). Given the rationality assumption, such observance of conventions is expected of speakers and is said to guarantee, in most cases, the willingness of the participants in a linguistic exchange to co-operate and to accommodate each others' needs. In that sense linguistic politeness with its central notion of face constitutes one of the primary pre-requisites of successful linguistic communication. Furthermore, and crucially to any dialogue model, according to B&L, politeness rules can be said to govern linguistic variation of which the most obvious manifestation is the varying degrees of indirectness with which the speakers choose to express their messages.

B&L proposed two main strategies (see Fig.1) to represent the social conventions that speakers use to make appropriate linguistic choices: the *On-record* and the *Off-record* strategies. They further split the On-record strategy into a strategy which does not involve redressive language and two sub-strategies which do, and which are aimed at Positive and Negative Face, respectively. Since redressive language typically involves linguistic indirectness, accommodating for the Positive and Negative Face can also be linked explicitly to linguistic indirectness.

1) Direct strategy: To a perfect stranger: *Give me some money now*!
2) Direct strategy with redress
   a) To a friend: *Look, I know you're broke just now, but I really need you to give me some money.*
   b) To a stranger: *I'm terribly sorry sir, but I'm short 10 cents for my ticket. Could you spare some change, please?*
3) Off-record strategy: To a lunch partner after lunch: *Drat! I left my purse at home.*

Figure 1: Examples of Brown & Levinson's strategies

Walker, Cahn & Whittaker (1997) combined B&L's theory with computational work on speech acts from AI (Allen & Perrault 1980, Cohen 1978), and devised algorithms that enable artificial agents to engage in linguistic style improvisation (henceforth LSI) in an interactive story. An important basis for their algorithms is speech act theory's distinction between the underlying intention of a speech act and the surface forms that can be used to realize that speech act. For example, the REQUEST-ACT speech act has an underlying intention of the speaker getting the hearer to do a particular action, but this intention can be realized by a variety of *surface* speech acts, i.e., by particular sentential forms such as declaratives, interrogatives, or imperatives.

Our Approach

In our work, we have also built on this basic notion that there are many different realizations for a particular intention, but we found several problems in trying to apply the LSI algorithms to tutorial dialogue. First, the work of B&L is not entirely applicable to tutoring: language produced in tutoring circumstances is governed by different norms than the language of normal conversation as described by B&L (Person 1995; Porayska-Pomsta 2003). For example, tutors do not tend to offer gifts or information to students as a way of fulfilling their needs, nor do they tend to apologize for requesting information from them. Thus, some of B&L's strategies simply do not apply to educational contexts, and others require a more detailed specification or complete redefinition. Specifically, the strategies need to be adapted to reflect both the pedagogical concerns such as the appropriate level of *content specificity* (how specific and how structured the tutor's feedback is with respect to the answer sought from the student), as well as motivational concerns which determine the appropriate level of *illocutionary specificity, i.e.,* how explicitly accepting or rejecting of the student and his answer the feedback is (Porayska-Pomsta 2003).

Second, the LSI operationalization is problematic for several reasons. The distance and power variables are relatively fixed for the tutorial context by the tutor-student roles that the interlocutors are playing. Thus, the rank of imposition accounts for all of the variation in realization. The LSI formulation would too narrowly constrain the range of utterances that could be produced. In our corpus, we have utterances from all categories, from on-record to off-record. Another problem is that the ranking of speech acts used in LSI does not transfer to tutoring. Finally, the LSI approach accounts for the generation of single speech acts, whereas tutorial feedback may include several speech acts in one utterance.

To generate feedback for tutorial dialogues, we had to:

- identify the situational factors which are important in tutorial interactions, and determine how they impact on the settings for the variables that determine face threat,
- develop methods for determining the values of situational factors,
- determine feedback strategies and their realizations from a corpus of tutorial dialogues, and

- develop discourse plan operators and realization rules for presenting tutorial feedback to the student that are sensitive to the values of the situational variables.

In our approach, we recognize the crucial role of facework in successful language production, but we redefine B&L's Negative and Positive face dimensions directly in terms of:

- Autonomy: Tutors should let students do as much of the work as possible (determination of the appropriate level of content specificity).
- Approval: Tutors should provide the students with positive feedback as much as possible (determination of the appropriate level of illocutionary specificity).

Values for autonomy and approval (henceforth A&A) are used to choose both the high-level tutorial strategy and the linguistic form for realization.

Relevant Situational Factors

We studied a corpus of human-human tutorial dialogues in the domain of basic electricity and electronics (BEE) in order to catalogue linguistic patterns of tutor feedback in terms of <aut, app> values. The dialogues along with the relevant educational literature and informal interviews with professional teachers were also used to determine a possible set of contextual factors relevant to teachers' decisions. In order to determine how tutors perceive the importance of the individual factors to their feedback decisions, we performed a study in which teachers were given situations, as characterized by combinations of values of the factors shown in Fig. 2. For each combination, the teachers were asked to rank the situational factors to indicate the relative importance to their feedback decisions (Porayska-Pomsta, Mellish & Pain, 2000).

Student-oriented factors
- student confidence
- student interest (bored/motivated)

Lesson-oriented factors
- time left for lesson
- amount of material left to be covered
- difficulty of material
- importance of material

Performance-oriented factors
- correctness of student's previous answer(s)
- ability of student

Figure 2: Situational Factors

The results of the studies were used to inform the design and implementation of the situational model. Specifically, a Principle Components Analysis allowed us to group the factors according to how they relate to one another, while teachers' written comments and post-hoc interviews allowed us to determine their possible relation to the A&A dimensions.

We built a Bayesian network that combines evidence from the factors to compute values for A&A for each tutor turn. The structure of the network reflects the relationship of factors as determined by the study. The network consists of four levels: (1) the input level represents the situational factors, their values and their relative importance; (2) the goal level represents Guidance-Oriented-Goals and Approval-Oriented-Goals; (3) the voting level combines the recommendations of the goals at level 2; and (4) the final outcome nodes represent Autonomy and Approval. At the second level, the values of the student- and lesson-oriented factors evoke specific goals that can be interpreted either in terms of guidance or approval. The relative importance of the factor values is used to calculate the strength with which these goals are evoked. The performance-oriented factors are used to modify the outcome of the goal recommendations on the third level. For a more detailed description see Porayska-Pomsta (2003).

"No, that's not right." tell incorrect <1.0, 0.1>

"Are you sure that this is the right way to de-energize the circuit?" question certainty: <0.8, 0.4>

"Not quite, why don't you try again?" tell incorrect plus gauging question <0.6, 0.4>

"Removing the wire does not de-energize the circuit. disconfirm effect <0.4, 0.1>

"If you remove the wire, then this will break the circuit but does it de-energize it?" question effect <0.3, 0.5>

"Isn't this breaking the circuit rather than de-energizing it?" question validity <0.2, 0.3>

Figure 3: Example feedback with A&A values

Generating Feedback Based on A&A

To generate appropriate feedback, we devised feedback strategies sensitive to the values computed for A&A in the current situation. The discourse planner uses the A&A values when choosing feedback strategies, and the surface generator uses them to choose among applicable realization rules. Fig. 3 shows 6 different feedback strategies that can be used when the student has made a particular error (removing a wire to de-energize the circuit). The name of each strategy is underlined, and the A&A values associated with the strategy are given in angle brackets. The discourse planner and realization components choose the operator and realization rules with the *minimal distance* between their <aut, app> value/range and the target <aut, app> pair, as indicated by the situation modeler (BEESM).

Overview of BEETLE System

BEETLE, our Basic Electricity and Electronics Tutorial Learning Environment, aims at improving student learning by prompting students to construct knowledge for themselves. The response generation module computes appropriate tutorial moves and synthesizes tutorial feedback as text and GUI actions. In order to combine the ability to plan tutoring activities with the ability to handle unexpected responses, interruptions and failures, BEETLE uses a three-level architecture inspired by work in robotics (Zinn, Moore, & Core, *in press*). Its three layers are: a *deliberative planner* that projects the future and anticipates and solves problems (top layer); a *sequencer* or *plan execution and monitoring system* that performs adaptive on-the-fly refinement (middle layer); and a *controller* or *perception/action system* that interprets student actions and performs primitive tutor actions (bottom layer). The deliberative planning and execution monitoring modules are implemented in the Open Planning Architecture (O-Plan) (Currie and Tate 1991). BEETLEGEN is the part of the bottom layer that performs surface realization, and is described in more detail below.

The complete BEETLE system also has: a robust NLU module based on the CARMEL toolkit (Rosé 2000) which translates the user's typed input into a logical form; a dialogue manager built using the TRINDIKIT dialogue system shell (Larsson and Traum 2000) which maintains the system's information state (including the dialogue history) and encodes rules of conversation; a domain reasoner encoding the system's knowledge about the subject matter being taught (implemented in Loom, a description-logic based knowledge representation and reasoning engine, see *http://www.isi.edu/isd/LOOM/LOOM-HOME.html*; a GUI for displaying a virtual laboratory, hypertext lessons and multiple choice questions, and a situation modeling component, which we describe next.

Situational Modeling: BEESM

BEETLE observes student actions and passes the relevant information, encoded as messages, to the situational modeler (BEESM). Messages can encode information of the following kinds:

- Aspects of the domain knowledge: difficulty and importance of the material;
- Temporal aspects: time allowed for a given lesson, time elapsed from the start of a lesson, amount of material left and material covered;
- Aspects of student behaviour: answer correctness and hesitation.

A set of diagnostic rules then processes these messages, interpreting them as situational factors having specific values. Then the relative importance of the situational values is determined using case-based reasoning. The case base implements the results of our teacher study. The values of the situational factors along with their relative importance is then used as evidence for the input nodes of our Bayesian net. Probabilistic inference is then used to determine the A&A values.

For example, when BEETLE sends a message of the form *action_correctness(<action>,incorrect)* to BEESM, the diagnostic rules identify this observation as the value for the performance-oriented situational factor "correctness". In some contexts, as maintained by the rule engine, BEESM may also infer values for the factors "student interest" and/or "student ability". For instance, if the student incorrectly attempted the same action several times, the inferred evidence for the node "student ability" will be the value *low*. Case-based reasoning is used to compute the relative importance of correctness and potentially of other situational factors. The output of the Bayesian reasoning, <aut, app> values, informs (strategic) discourse planning and (tactical) sentence realization.

Strategy Selection: Discourse Planner

For brevity, we describe O-Plan's operationalization of A&A values with an example. Fig. 4 depicts one of the many O-Plan operators for the generation of tutorial feedback when the student has incorrectly performed a step in a procedure.

```
schema do_supply_feedback_disconfirm_effect;
    vars      ?step            = ?{satisfies listp},
              ?aut, ?app       = ?{satisfies numberp},
              ?sact            = ?{satisfies listp};
    expands   {supply_feedback ?step};
    only_use_for_effects {action_correctness ?step} = correct;
    conditions
       only_use_if  {answer_correctness ?step} = incorrect,
       compute {beesm (get_aut_app)} = {?aut, ?app},
       compute {nearest_neighbour ?aut 0.4 ?app 0.1} = true,
       compute {tis (get_last_utt student action)} = ?sact;
    nodes   1 action {take_turn},
            2 action {reject_action},
            2 action {disconfirm_effect ?step ?sact},
            3 action {give_away_turn};
end_schema
```

Figure 4: A Discourse Plan Operator

This operator aims at achieving the effect {action_correctness ?step} = correct, and is applicable when the conditions are satisfied. The first condition indicates that the operator may be used when the student's action is incorrect. Other conditions query external knowledge sources (e.g., BEESM and TIS, the agent maintaining the information state) to obtain instantiations for O-Plan variables (prefixed with '?'). Of interest here is the nearest_neighbour condition, which discriminates between various supply_feedback operators. It checks whether the distance between the computed A&A values and the given constants is minimal. If the conditions are satisfied, the operator is expanded into a sequence of more

primitive steps. In our example, supply_feedback unfolds into four sub-steps: take the turn, reject the student's action, disconfirm the correctness of the student action, and then give away the turn. Once a discourse plan has been found, BEETLE's execution and monitoring component selects a sequence of plan steps for execution, transforms the O-Plan representation into XML (*cf.* Fig.5), and passes it to BEETLEGEN for realization.

```
<input>
  <take_turn move-id="9"/>
  <reject_action move-id="3"/>
  <disconfirm_effect move-id="17">
    <de_energize>
      <I-CIRCUIT1/>
    </de_energize>
    <remove>
      <I-CIRCUIT1/>
      <I-WIRE-1451/>
    </remove>
  </disconfirm_effect>
  <give_away_turn/>
</input>
```

Figure 5: XML input to the text realizer

Linguistic Realization: BEETLEGEN

BEETLEGEN transforms a set of dialogue act specifications into one or more surface realizations using XSLT. The input and output data structures are represented in XML, and the transformation is performed by means of a pipeline of stylesheets (Clark, 1999). Our pipeline model is similar to that of Wilcock (2003), but we extend his approach by adding a distance function, subcategorization, and context-dependent pronominalization and variation. We discuss the most important characteristics of our implementation. An XSLT processor works by traversing an input XML document top-down and left-to-right, writing XML elements and attributes into the output tree. At each node of the input tree, the XSLT processor chooses the best matching XSLT-template and executes it. This makes processing deterministic and efficient, but also makes it difficult to perform search. However, by encoding appropriate match conditions (in the XPATH language), we can reduce the risk of making wrong choices.

The realizer works by building an initial text structure, and then successively refining and filtering it. Fig. 5 shows the input the realizer will be given when the supply_feedback operator above has been expanded; the move-ids are identifiers for speech acts in the Information State. This input will ultimately generate the utterance *"Removing the wire does not de-energize the circuit."* (*cf.* Fig. 3). When traversing the input XML tree, we first encounter two elements with no children: take_turn and reject_action. Elements with no children are mapped directly to phrases by simple XSLT templates. In dialogue, it is common for turn-taking and acknowledgement moves to be performed implicitly by subsequent speech acts, thus avoiding utterances like: *"Umhm. No. Removing…"* We use XSLT templates that do not write anything to the output tree to block verbalization of such moves where appropriate. Blocking the realization of input speech acts in this way implements the subsumption principle of Stent (2002).

For example, in the current implementation we block explicit *acknowledgment* moves if there are also *direct* or *assert* moves in the input. Likewise, *reject_action* moves are blocked in many situations, for example if they are followed by a *diag_query*, *disconfirm_correctness* or *tell_s_incorrect* move. This is because the latter moves realize more fine-grained and appropriate reactions to a student's mistake than the more general *reject_action* move, which could negatively affect the tutor's strategy by adding *"No"* to output that was intended to be indirect.

```
<xsl:template match="disconfirm_effect">
  <node syncat="sent">
    <obligatory>
      <xsl:apply-templates select="*[2]" mode="vp-gerund"/>
    </obligatory>
    <word phon="does"/> <word phon="not"/>
    <obligatory>
      <xsl:apply-templates select="*[1]" mode="vp-inf"/>
    </obligatory>
    <punct phon="."/>
  </node>
</xsl:template>
```

Figure 6: XSLT-template building syntactic structure

The *disconfirm_effect* element, which contains propositional content in the form of child elements, is realized by means of a generation template into which the propositional content is inserted (*cf.* Fig. 6). This is done by first generating a sentence frame containing the canned text *"does not"* and the sentence final full stop, and then recursively calling xsl:apply-templates with the children of the matched *disconfirm_effect* element. Thus, our realizer mixes template-based and rule-based processing. The additional XSLT calls make use of XSLT-mode declarations which serve to distinguish subsets of XSLT templates. We use XSLT templates in different modes to generate different output trees for the same input element, depending on the syntactic features that are required. In Fig. 6, we require a gerund for the first part of the sentence and an infinitive for the second.

In the remaining stylesheets of the pipeline, we perform several operations. To prevent incomplete output sentences, we check whether the obligatory elements introduced by rules such as the one in Fig. 6 have any child elements. The next step of the pipeline makes basic decisions about pronominalization for multi-sentence turns. For this, we use XPATH expressions in the match

conditions of XSLT templates to test whether the current NP node has the same semantics as an NP node of the previous sentence. At this stage in the pipeline, with the exception of small pieces of canned text, the generator has not yet produced any final word forms. This is because it must wait until pronominalization decisions have been made. After this is done, a lexical lookup stylesheet produces fully inflected word forms by matching syntactic features specified in its input XML tree. The stylesheet contains default morphological rules as well as exceptions to those rules. The ordering of the XSLT templates guarantees that the exceptions take precedence over the general rules. The final stylesheet extracts lexical items from the generated tree structure, upcases sentence-initial words, applies rules of punctuation and controls the insertion of whitespace.

There is other processing that we have not discussed above: some XSLT templates use <aut, app> values to generate different verbalizations, and some access BEETLE's Information State to obtain information about previous word choices in order to avoid repetition.

Evaluation & Future Work

We performed an evaluation of the situational modeler and the linguistic strategies associated with A&A values. Four experienced BEE tutors took part in the study. Each was presented with twenty different situations in the form of short dialogues between a student and a tutor. Each interaction ended with a student answer that was either incorrect or partially correct. For each situation, the participants were provided with three possible tutor responses to the student's answer and were asked to rate each of them on a scale from 1 to 5 according to how appropriate they thought the response was in a given situation. They were asked to pay special attention to the manner in which each response attempted to correct the student. The three types of responses rated included: a response that a human made to the given situation, the response recommended by the situational modeling component, and a response that the model was less likely to recommend for the same situation.

We performed a t-test to determine whether there was a significant difference between the three types of responses. The analysis revealed a significant difference between human follow-up responses and the system's less preferred responses *($t(19) = 4.40$, $p < 0.001$)*, as well as a significant difference between the system's preferred and the system's less preferred responses *($t(19) = 2.72$, $p = 0.013$)*. Finally, and most encouraging, there was no significant difference between the ratings of the human responses and the system's preferred responses, *($t(19)=1.99$, $p=0.61$)*. This preliminary analysis indicates that the model's choices are in line with those made by a human tutor in identical situations (for more detail, see Porayska-Pomsta, 2003). This bodes well for the implementation, and in future work, we will evaluate the situational model and generation capabilities in the context of the full BEETLE system.

References

Allen, J.F. and C. R. Perrault. 1980. Analyzing intention in utterances. *Artificial Intelligence* 15:143–178.

Bloom, B. S. 1984. The 2 Sigma problem: The search for methods of group instruction as effective as one-on-one tutoring. *Educational Researcher* 13:4-16.

Brown, P., and S. Levinson. 1987. *Politeness: Some Universals in Language Use*, Cambridge University Press.

Clark, J. 1999. XSL Transformations (XSLT), Version 1.0, W3C Recommendation. http://www.w3.org/TR/xslt.

Cohen, P. R., 1978. On knowing what to say: Planning speech acts. Technical Report 118, University of Toronto.

Currie, K. and A. Tate. 1991. O-Plan: the Open Planning Architecture. *Artificial Intelligence*, 52:49–86.

Larsson, S. and D. Traum. 2000. Information state and dialogue management in the TRINDI dialogue move engine toolkit. *Natural Language Engineering*, 6(3–4):323–340.

Fetzer, A. 2003. 'No Thanks': A socio-semantic approach. *Linguistik* (14): 137-160.

Fox, B. 1993. *The human tutorial dialogue project: Issues in the design of instructional systems*. Lawrence Erlbaum.

Lepper, M.R. and R.W. Chabay. 1988. Socializing the intelligent tutor: Bringing empathy to computer tutors. In Mandl and Lesgold (Eds.), *Learning Issues for Intelligent Tutoring Systems*, 114-137. Springer.

Person, N. K., R. J. Kreuz, R. A. Zwaan, and A.C. Graesser. 1995. Pragmatics and pedagogy: Conversational rules and politeness strategies may inhibit effective tutoring. *Cognition and Instruction* 2(13), 161-188.

Porayska-Pomsta, K., C. S. Mellish, and H. Pain. 2000. Pragmatic analysis of teachers' language: Towards an empirically based approach. *Proc. of the AAAI Fall Symposium on Building Systems for Tutorial Applications.*

Porayska-Pomsta, K. 2003. Influence of situational context on language production: Modeling teachers' corrective responses. PhD thesis, University of Edinburgh.

Rosé, C. P. 2000. A Framework for Robust Semantic Interpretation. In *Proc. of NAACL.*

Stent, A. 2002. Conversation Acts–Model for Generating Spoken Dialogue Contributions. *Computer Speech and Language, Issue on Spoken Language Generation.*

Wilcock, G. 2003. Integrating Natural Language Generation with XML Web Technology. In *Proc. of the Demo Sessions of EACL-2003,* pp. 247-250.

Walker, M.A., J. E. Cahn and S. J. Whittaker. 1997. Improvising Linguistic Style: Social and Affective Bases for Agent Personality. *Proc. of the First International Conference on Autonomous Agents.*

Zinn, C., J. D. Moore and M. G. Core. In press. Intelligent information presentation for tutoring systems. In *Intelligent Information Presentation*. In Stock & Zancanaro (Eds.), Kluwer.