Interaction Informed Design of User Modeling for Rapport

William Liu
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA
willixl@cmu.edu

ABSTRACT

Current research and discussion on user modeling either focus on backend infrastructure or on the possibilities they provide in the scope of human-system interaction. But many do not strongly relate and demonstrate the dependencies between these two aspects of designing and building a user model. Our paper attempts to present the process of building a user model for an existing system that builds rapport as an algebra tutor. We use information of the system's infrastructure as well as the goal of the system to guide our design. We explicitly identify the problems with the current system and the solutions that adding a user modeling subsystem would bring, specifically matching each solution to a target problem. Then based on the requirements of our design, construct a backend infrastructure for storing the necessary user modeling data, with the capability to update the database the necessary amount for the task that our specific user model is attempting to accomplish. This way we can ensure that our user model is space and operation efficient, and is able to solve the problems it was designed to solve, but not unnecessarily over-engineered.

KEYWORDS

user modeling, rapport building, virtual agent, virtual tutoring

1 INTRODUCTION

Dialogue systems that incorporate social aspects into their speech and interaction can many times benefit from some form of user modeling. To improve social interaction is to improve the rapport between the interaction's participants [13]. Building rapport, and thus improving social interaction, can be achieved through things such as mutual attentiveness, face management, and commonality, etc [12,13,14,15]. There are already many dialogue and other systems that implement user modeling to achieve adaptive systems. Systems such as route recommenders that omits non-critical information [6] or communication systems that entrain to the user communication over time to communicate better [9]. In the first case, heavy attention is given to the backend infrastructure of how the linguistic data is stored and computed when giving out route information. In the second example, it is quite the opposite. Focusing much more on the effects of a user model for allowing their system to entrain to a human. We will be discussing how to fit a user model into an already established system that has the goal of being a better algebra tutor by building rapport with the student. Then we will identify the gaps in the current research, namely that most papers and discussions focus on one of either backend infrastructure or the benefits of a user model. We seek to fill that gap by discussing how the desired interaction informs the design of the infrastructure and how the infrastructure influences the interaction. The second gap being that there are not any attempts at user modeling specifically for a system that builds rapport as well as having an on-task goal.

We attempt to address these gaps by prototyping our own user model for a rapport building algebra tutor system we already have in place called RAPT. The user model prototype we have built has not yet been implemented into the RAPT system and thus has not yet been piloted. However the gaps that we are attempting to fill are not directly validated through piloting and testing. Rather the first gap that we are trying to fill is to bring the discussion of the user model design process into the research domain of user modeling implementations, which is not validated through pilot testing. And the second gap can be validated through direct unit testing with the user modeling subsystem without having to integrate it into the full system as of yet.

2 BACKGROUND

2.1 The RAPT System

The RAPT system performs user tests on two distinct trial versions of algebra tutoring: (1) straightforward tutoring that does not involve any social aspects and stays on task for the entire duration of the trial period, (2) tutoring that involves social moves when delivering information and hints, dependent on the situation, and will engage in off-task conversation in the middle of on-task algebra problems.

The sequence of major events for the social version of the RAPT system begins with a few minutes of introductions and 'get to know you' questions, such as asking about hobbies and interests. This initial conversation is quite awkward, both by the nature of the interaction type as well as how the system currently handles the initial interaction. However, the discomfort as well as the anxiety the student feels from being in a new learning environment with a stranger makes it easier for the RAPT system to initially model the student. In a more anxious state, it is easier to computationally parse for t both language and syntax, as well as recognize personality and sentiment [1,2].

After initial introductions, the tutor moves to on-task algebra tutoring. The algebra problems are classified by their knowledge components, or KCs. KCs represent different components that define a problem. Some example KCs include: fraction coefficients, variables in the denominator, negative coefficients, etc. KCs are assigned generally with increasing difficulty such that KC 2 is generally more difficult than KC 1 for a new student to solve. The RAPT system is currently able to estimate the student's mastery of each KC on a scale of 0 to 1. Where 0 means the student has 0 chance at solving a problem and 1 means the student will one hundred percent be able to solve a problem. At this stage, the social version of the RAPT trial will converse with different social moves such as violating social norms, eliciting self disclosures, and so on. A Wizard-of-Oz (WoZ) setup is used, where the WoZ chooses responses and social moves from a sequence of actions detailed by a finite state machine. However while the current system can make social moves, these moves are not informed from any empirical user statistics, rather the WoZ may choose social moves based on student performance or shared experiences. The finite state machine may also give hints and comment on the student's performance over time, potentially even bringing in previous hints to help the student out on a similar problem that they are currently struggling with. The subsystem to automatically capture this performance and shared experience (etc.) data is still currently lacking.

At this time, trials are only run once per student. Which means any choice to reference a shared experience or perform any other social move is based on data within a single trial. There are potential plans to scale the trials to having more than one tutoring session per student.

2.2 Language Parsing

The focus of this user modeling prototype is on the process of gathering, storing and using agent-student interaction to inform choice of social moves. To fill in the missing components of the prototype user model, Google Cloud Platform's Natural Language API [3] is used for natural language parsing. Using the API, three separate characteristics of dialogue were used for the user model.

2.2.1 To recall back to certain events or to discuss certain hobbies, all of the student dialogue is parsed for its *entities*. An entity is defined as proper nouns such as public figures, landmarks, etc. [3]. The Natural Language API parses for such entities in the student dialogue and returns the entities themselves as well as the broad category that they fall into (as defined above).

2.2.2 The *salience* of an entity describes how important and relevant that specific entity is in the context of the entire sentence or utterance. The salience of an entity is scored between 0 and 1, where 0 means the entity has no relevance in the sentence and a 1 means that the entity is the sentence.

2.2.3 Once salient entities are captured, they can be analyzed in context for the student's *sentiment* regarding those entities. Sentiment is the attitude of the student when speaking about entities in context of the sentence. Sentiment is scored on a scale of -1 to 1, where -1 means they absolutely hate the entity, 1 means they absolutely love the entity, and 0 means that they have an absolutely neutral attitude towards the entity.

3 RELATED WORK

Looking at prior literature as well as the goal for creating a user model for the RAPT system, we need to immediately define what it means to be a user model and what a user model is meant to accomplish. According to Kass at the high level "a user model is the knowledge about the user, either explicitly or implicitly encoded, that is used by a system to improve the interaction" [4]. Kobsa further expands upon this definition by adding that the system should also have some model to track the improvements that have been made [5].

In the case of many papers in the domain of user modeling, improving interaction means to adapt speech over time by being more or less colloquial or entraining to the human speaker [6,9,10] or providing better information by wrapping the output information based on who was asking [7,8]. Wrapping the output information could mean giving better context for the information given. For example, for a virtual advising system, instead of telling an advisee simply to take a course, context could be given that the course in question is a graduation requirement, based on information about the student's major [7]. While van Beek's system, though still containing some black boxes around the computations for generating cohesive wrappers around all of the output information, can create a personalized experience, the findings and discussions presented are centered largely around data aggregation and natural language techniques. However, the databases that they use are somehow able to just contain all of the data that they need for all of their interactions, without much discussion on edge cases or how they got to the dataset that they have. Also van Beek's system is adaptive only with the information that the system presents, but does not adapt the social interaction between the system and the student in any way.

On the other hand, there are the papers that discuss how to adapt the actual dialogue patterns and colloquialness of the agent. They don not delve much into how they store all of their data, but rather focus on the human-agent interface. For example in an app that gives bus directions, a new user of the app will have the agent speak to them very formally and give as much information as they can about the routes, how to get to them, and when to watch out for their stop. Whereas for veteran users, it skips many of the formalities and just gives them the bus timings, assuming that the user does not need the excess information [6]. Or, in Kanashiro's paper, an agent entrains to the speech patterns of the human they are conversing with and slowly adopts their way of typing [9]. In both of these systems, the goal is for the agent to get better at conversing with the human, either by not overloading the amount of useless information given each and every time, or by providing a realistic progression of conversation since entrainment naturally occurs between two humans [11]. Here a gap arises in which papers that focus on backend infrastructure [7,8] don not present that infrastructure in the context of the problem that they are attempting to solve. They also create 'black boxes' around the computations they use to facilitate the interactions, assuming that well-constructed sentences are just a given. On the other hand, papers that discuss how a user model can change the human-system interaction in their specific case [6,9,10], they create 'black boxes' around the infrastructure that would gather and maintain the data required for such goals.

With requiring all of this data gathering, comes questions of: where to gather this data from, and when to gather and update data. Depending on the information needed for the user model to be effective, data points may be very easily inferred such as primarily gathering data about how long a user has been using a particular application [6] or having all of the data scraped from a complete repository of student information [7]. Of course the obvious, but most difficult to parse from, source would be from the user-system interaction itself. Data points such as the user's vocabulary, dialogue patterns, and tone of voice [10,11] can all be taken as data points. Data such as a repository of courses and student information need to be gathered only once and updated infrequently, such as when courses change and when students graduate. The time-dependent data such as how many times a user has interacted with a system needs to be updated at regular increments that are fairly easy to determine based on how often the designers think warrant a change in the system. Dealing with entrainment requires significantly more updates to the user model. Updates can occur on every utterance from the user or large enough differences in current versus stored utterance data. The same gap arises in which there is not much discussion on when to update data for agents that are attempting to build relationships, by focusing on rapport.

4 METHODOLOGY

4.2 Identifying Problems in Current System

The RAPT system always starts off its sessions with a get to know each other session and will occasionally throw in more social banter during the tutoring session as well. These bursts of off-task conversations do not have to do with algebra at all. Looking a the human-human dataset, sometimes long discussions about a singular topic is maintained throughout a session and other times different topic pop up as the participants are learning algebra. Since the RAPT system uses a WoZ system as well as not yet performing longitudinal studies, it is easy for the human controller to maintain a 'database' of the topic that have been discussed already in the session (as this is a fairly easy task for humans, as seen in the human-human dataset as well) and is able to very easily extrapolate from what has already been said to allow the virtual agent to elicit new and engaging self disclosures from the participant. So a user model would need to mimic that aspect of the current WoZ system. And to, in the future, expand the trials longitudinally, running trial after trial until a student has actually mastered the subject of algebra (every tutor's dream), the user model must also be designed to handle large amounts of data if necessary.

Switching to on-task user modeling: A powerful tool at a tutor's disposal is a student's own experiences and abilities. When faced with a similar problem as one that has already been seen before, but perhaps asked in a different manner or mixing in other elements of a subject, students sometime need just a hint telling them to think back to a previous problem and use the methods from then to think about the problem at hand. For example, if fraction coefficients were covered either an hour ago or last week, and now the student is seeing variables in the denominator for the first time, the tutor can give a quick nudge that these two problem types can be viewed in the same way and take similar approaches to solving. Again the WoZ System can currently keep up with this task as a human is able to remember the past problems that the participant has attempted and/or struggled with.

However even for a WoZ human, running many participants and adding in multiple trials per participant would be a difficult. A human tutor or WoZ simply cannot keep up with so many social interactions and keeping up to date on the conversations. Both in remembering all of the social interactions, keeping them ordered by student and temporally as different tutoring sessions take place. As well as in the mastery level of each student over time, again both keeping the records straight between each student and temporally across multiple tutoring sessions over time. A user model would be able to easily solve the record keeping problem, and it must also be able to begin to take the place of the WoZ system currently in place.

4.2 Matching Solutions to Problems

To move towards a fully autonomous system that can maintain the current WoZ's ability to give a participant the feeling that the agent is learning more about a student and getting more familiar with that student, the proposed user model would need to keep a running log of what topic have been discussed and know how to continue discussing them. We decided to write the backend infrastructure of this prototype in Python and exporting all of the data out into various CSV files, classified by who and what it is modeling: the student/agent and on-task/off-task. Python was only chose as it is a powerful rapid prototyping language. Python only has basic data structures, so the rest of the discussion on data storage will mainly be on what to store and why, as well as when the user model should be looking to store data. We fully anticipate that a real, ready-to-integrate version of the system will not be written in Python. This means that the log must store and maintain the entities that have been mentioned throughout the course of all of the social dialogue. In our prototype, we split the social, off-task modeling into two sections: the initial interaction between the student and the agent and when the student comes back for subsequent trials in a longitudinal study. In the get to know each other period of the trial, the user model is actively eliciting as many self disclosures from the student as possible and storing all of the entities uttered by the student. Those are then filtered by salience so that very obviously non-salient entities, mostly corollaries to what was actually being discussed, were discarded to prevent edge case questions and discussions later on. In this stage, the agent may ask about the student's hobbies, interests, etc. And as those entities are being recorded, the sentiment of each of those are recorded as well as the rapport rating of the interaction at the time the entity is uttered as well as the change in rapport. The sentiment data can then be used to allow the agent to either side with or with against the student's sentiment. Such that there will be an agent model that exists alongside the user model, giving a more realistic feel to the conversation and so that the agent dialogue is consistent between trials once longitudinal studies begin. Switching over to the subsequent trial version of the social user model, the same database and storage structure is used. This time, the agent will go through the same introduction and social banter section as in the initial interaction. However instead of just eliciting self disclosures from a generic list, it will choose, currently based purely on probability, to follow up on entities discussed in the previous sessions. So if, in the first session, the student discloses that they have a soccer game coming up, in the next session the agent may choose to ask about the soccer game.

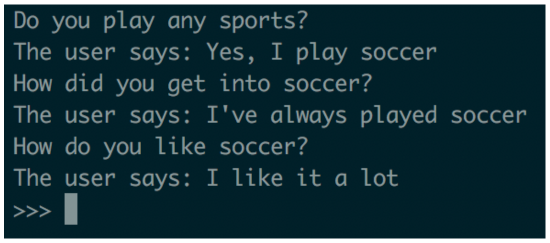
Currently there is no explicit sequence for how to switch to social banter in the middle of solving a problem, however it should look similar to how the beginning of non-initial sessions may look and generate dialogue using information from the same database.

The approach to solving the problems in the on-task domain of the RAPT system would take a very similar approach. Whenever the student begins on a new problem, the knowledge components, or KCs within that problems are taken down synonymously to how the entities functioned in the social version of the data structure. Again the sentiment, rapport level, and change in rapport are recorded. Additional parameters that are stored are the hints that are given by the agent, if any and the student's mastery level of the KC after the problem. Again, regarding sentiment the agent can either agree or disagree with a student on the feeling towards a specific problem type. In this case, whether or not they like that problem type. For example if the student hates fraction coefficients, the agent can agree and subsequently build rapport based on their mutual agreement of their hate of fractions, or the agent can disclose that they actually like fractions and subsequently build rapport by using their love of fractions to help the student. These sentiment agreements or disagreements are also stored in an agent model database. This data can then be accessed later when the student runs into another problem that has the fraction coefficient KC. The hints given prior, if any, can be given again. If none, the agent can still bring up the fact that the current problems bears resemblance to a previous problem and can prompt the student to make the connection. The on-task data can also be used for social interaction such as praising. If a student had been struggling with specific KCs for some time but was able to solve a new problem with that KC, that could signal for the agent to praise the student for their improvements.

When referencing to shared experiences, the natural inclination would be to reference the experiences that occurred the most recently. However that may not always be the case. Since the goal of RAPT is to build rapport, perhaps it may be best to reference interactions that had the highest rapport rating or the highest positive change in rapport [12,13,14]. The current system has functions in the backend that will re-sort the data based on whichever parameter, in either increasing or descending order (alphabetically for the entity and hints). This way when piloted, the sorting of different parameters can be experimentally tested for their ability to build rapport between the human-agent interaction.

5 CONCLUSION

We identified two gaps in the current user modeling domain. The first is that many user model designs in current research discussions don't very explicit about how their interactions and system design are influenced and informed by each other. In our methodology, we were very explicit about how the data structures we chose and what we chose to store came from singular problems in our system that we had identified. The second is that previous research has not looked into better rapport building by using a user model in a system. While we have not integrated our prototype user model into the full RAPT system yet, we are able to perform small scale unit tests to validate that the functionality that our original design was created to do performs as expected.



```
Do you play any sports?
The user says: Yes, I play soccer
How did you get into soccer?
The user says: I've always played soccer
How do you like soccer?
The user says: I like it a lot
>>> 
```

FIGURE 1: TESTING INITIAL INTERACTION

To the best of our knowledge, this is the first paper to deliberately break down the requirements of a user model for a system and match infrastructure specifications to each identified requirement. Also to the best of our knowledge, this is the first paper to explicitly implement a user model for rapport building. These findings are important because it provides a new design methodology to creating user modeling systems: by looking at the goals and not over-engineering inefficient solutions. It also shows that the user modeling aspect of a WoZ in a rapport building system *can* be replaced by a user modeling subsystem instead.

There are many future works and discussions that can come of this paper. Since the user model discussed in this paper is still an early stage prototype, a clear next step would be to fully build the system using a proper backend programming language and real space-efficient data structures and storage servers. Once a non-prototype version of the system has been built, a clear next step would be then to integrate the user modeling subsystem into the current RAPT system and be able to pilot RAPT without the WoZ controlling any of the user modeling with students. Additionally, we can also prepare the user modeling subsystem for full capability to pilot for a longitudinal study should RAPT begin to do so. Another interesting aspect of the user model, specific to ours, to consider is that once the user's hobbies, preferences, activities, etc. have been stored, the current user model can only act on the specific entities that are stored. However a real conversation would expand from a single entity outwards. For example, if a student were to disclose that they are a fan of soccer, our implementation would directly ask 'how did you get into *soccer*?'. In real-life conversations, if the agent were to know a little bit about soccer, they could instead follow up the student's self-disclosure with 'what is your favorite team?' or 'do you play any other sports?'. This would require hierarchal understanding of the entities being stored in the conversations. For example we know that in soccer there are teams and players, and that soccer falls under the category of sports and physical activities. Having this feature in the implementation can give the agent a much more natural sounding sequence of questions and follow ups as well as be able to better fill out the user model database with more entities and their associated values in all of the parameters discussed prior. Finally one more future direction to consider would be also modeling non-verbal behaviors that contribute to rapport building. Just like how verbal behaviors are modeled now, we would most likely follow the same problem-solution matching design process to create the skeleton of the backend infrastructure.

A HEADINGS IN APPENDICES

A.1 Introduction

A.2 System Background

A.2.1 The RAPT System

A.2.2 Language Parsing

A.3 Literature Review

A.4 Methodology

A.4.1 Identifying Problems in Current System

A.4.2 Matching Solutions to Problems

A.5 Conclusions

A.6 References

ACKNOWLEDGMENTS

Thanks to Michael Madaio for giving such a fun first introduction to the research domain of virtual agents. And giving insightful discussions and guidance along the way. Also thanks to the other interns in the lab for keeping a positive research atmosphere, especially the other independent study students that are also chugging along with their research topics.

REFERENCES

[1] Mairesse, F., & Walker, M. (2000). Words Mark the Nerds: Computational Models of Personality Recognition through Language. 28th Annual Conference of the Cognitive Science Society, (May), 543–548.

[2] Mairesse, F., Walker, M. A., Mehl, M. R., & Moore, R. K. (2007). Using linguistic cues for the automatic recognition of personality in conversation and text. Journal of Artificial Intelligence Research, 30, 457–500.

[3] Google Cloud Natural Language API Documentation | Google Cloud Natural Language API | Google Cloud Platform, Google, cloud.google.com/natural-language/docs/.

[4] Kass, R., & Finin, T. (1988). MODELING THE USER IN NATURAL LANGUAGE SYSTEMS Robert Kass and Tim Finin, 14(3).

[5] Kobsa, A., & Wahlster, W. (1989). User Models in Dialog Systems.

[6] Komatani, K., Ueno, S., Kawahara, T., & Okuno, H. G. (2003). Flexible Guidance Generation using User Model in Spoken Dialogue Systems, (July), 256–263.

[7] van Beek, P. (1987). A model for generating better explanations. Proceedings of the 25th Annual Meeting on Association for Computational Linguistics -, 215–220.

[8] Carberry, S., Chu-Carroll, J., & Elzer, S. (1999). Constructing and Utilizing a Model of User Preferences in Collaborative Consultation Dialogues. Computational Intelligence, 15(3), 185–217.

[9] Kanashiro, I., Kobayashi, K., & Kitamura, Y. (2009). Entrainment in human-agent text communication. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5044 LNAI, 268–277.

[10] Iio, T., Shiomi, M., Shinozawa, K., Shimohara, K., Miki, M., & Hagita, N. (2015). Lexical Entrainment in Human Robot Interaction: Do Humans Use Their Vocabulary to Robots? International Journal of Social Robotics, 7(2), 253–263.

[11] Brennan, S. E. (1996). Lexical Entrainment in Spontaneous Dialog. Proceedings, 1996 International Symposium on Spoken Dialogue, ISSD-96, 41–44.

[12] Cicourel, A. V. (1974). Cognitive sociology: Language and meaning in social interaction. New York: Free Press

[13] Cappella, J. N. (1990). On Defining Conversational Coordination and Rapport. Psychological Inquiry, 1(4), 303–305. https://doi.org/10.1207/s15327965pli0104_5

[14] Zhao, R., Papangelis, A., & Cassell, J. (2014). Towards a Dyadic Computational Model of Rapport Management for Human-Virtual Agent Interaction BT - Intelligent Virtual Agents: 14th International Conference, IVA 2014, Boston, MA, USA, August 27-29, 2014. Proceedings. In T. Bickmore, S. Marsella, & C. Sidner (Eds.) (pp. 514–527). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-09767-1_62

[15] Tickle-Degnen, Linda; Rosenthal, Robert (1990). "The Nature of Rapport and Its Nonverbal Correlates" (PDF). Psychological Inquiry. 1 (4): 285–293. doi:10.1207/s15327965pli0104_1.