Understanding Speech in Interactive Narratives with Crowdsourced Data

Jeff Orkin and **Deb Roy**
MIT Media Lab
75 Amherst St.
Cambridge, MA 02139
{jorkin, dkroy}@media.mit.edu

Abstract

Speech recognition failures and limited vocabulary coverage pose challenges for speech interaction with characters in games. We describe an end-to-end system for automating characters from a large corpus of recorded human game logs, and demonstrate that inferring utterance meaning through a combination of plan recognition and surface text similarity compensates for recognition and understanding failures significantly better than relying on surface similarity alone.

Introduction

In today's world of speech-enabled technologies (e.g. *Siri*, *Watson*, *Kinect*), characters in games are begging for us to *talk* to them. Yet, understanding spoken natural language remains a challenge. The most robust speech recognition solutions, trained on enormous corpora, running on the cloud, can be hampered by background noise, microphone quality, speaking volume, speaking styles, or other factors. Even with perfect recognition, characters may find words or phrases unfamiliar or ambiguous.

Non-player characters (NPCs) require adequate *coverage* of all utterances they might be expected to understand. Thus, understanding language is subject to the same authoring bottleneck that plagues generation of behavior and dialogue. In the age of big data, where it is increasingly easy to record, store, and process data from players interacting online, we can revisit this problem.

We describe a system that leverages a corpus of thousands of recorded human interactions to not only address the coverage problem, but to also compensate for speech recognition failure by exploiting narrative context. Speech is *understood* by mapping player input to an utterance in our corpus -- a two-step process. Like a search engine, the system first retrieves a list of relevant utterances from the corpus, *semantically* similar to the input, possibly expressed with different words. Next, the

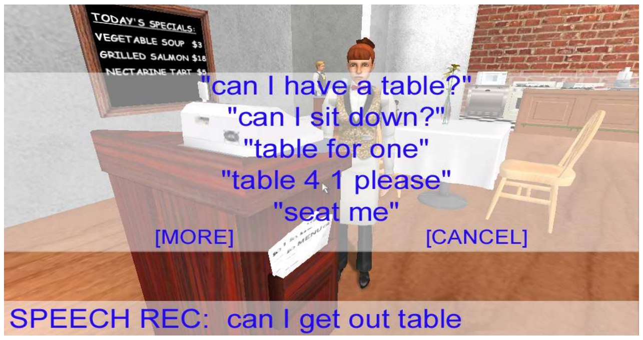


Figure 1: Intelligent interface compensates for misrecognition.

single best match must be selected from the retrieved list. Here we focus on the first step, and leave the second step to the player. We present an intelligent interface that dynamically populates a list of dialogue options based on speech input, and allows the player to select what s/he is actually trying to say. Including a human in the selection process allows us to measure the quality of our search results, in terms of the rank of the selected utterance, and the frequency of aborting, when the player can find no satisfactory option.

We have integrated this interface into *The Restaurant Game* (TRG), and observed human customers interacting with waitress NPCs. TRG is a long-term project working toward data-driven NPCs who understand and generate language by exploiting a large corpus of human demonstrations. We have previously described automating NPC-NPC interactions with statistical models (Orkin & Roy 2009). This paper describes and evaluates an end-to-end system for automating *human*-NPC interactions from annotated data, with integrated speech recognition.

We present results from an experiment comparing three different means of retrieving and filtering dialogue options, and find that combining text surface similarity with plan recognition is 18% more likely to find a relevant dialogue option than relying on surface similarity alone, and can often provide relevant options even in the event of complete speech recognition failure. Findings suggest that leveraging contextual knowledge provided by recorded demonstrations to compensate for language understanding failures can significantly improve the viability of speech interaction in interactive narratives.

Collective A.I. for Interactive Narratives

Collective A.I. refers to an end-to-end process for recording online games, discovering patterns in data, and automating data-driven NPCs. The intelligent interface supporting speech input relies on the same machinery that drives NPC behavior. We give an overview of the process, followed by a description of how the system supports human interaction. More details about data collection, annotation, and pattern discovery are available in previous publications (Orkin & Roy 2007; 2010). To date we have recorded 10,027 logs, and annotated 1,000 logs. This study evaluates humans interacting with an NPC driven by a 100 game subset of the annotated logs.

Crowdsourced Data Collection

TRG anonymously pairs humans online as customers and waitresses in a restaurant (using *Torque 3D* for Windows and OSX). Players can converse via typed text, and interact with 47 types of objects in the 3D environment through a point-and-click interface. Games generate text-based logs of time-coded actions, state changes, and utterances.

Data Interpretation by Humans

While powerful algorithms exist for learning patterns from data, we choose to rely on human interpretation. Our motivation for recording humans online is to capture the nuance and variety of behavior and language, subtleties that wash away statistically due to sparse data. By employing humans to interpret data, we can capture valid examples of interaction that may have only been observed in few games, or even only once.

We hire non-experts online to interpret and generate abstract representations of data. Annotators use browser-based tools (AS3/Flex) to view game logs as timelines of nodes representing actions and utterances, and tag them with four varieties of meta-data: *events*, *event hierarchies*, *causal chains*, and *references*. Annotators draw boxes around sequences of nodes to tag events, and draw bigger boxes around multiple events to tag event hierarchies. Events may contain actions and utterances, arbitrarily intermixed. We have 31 low-level events (e.g. `GET_SEATED`, `ORDER`, `PAY_BILL`), grouped into five higher-level events (e.g. `BEGIN_DINING`, `CONCLUDE_DINING`, `FULFILL_ORDER`). A domain expert defines the list of event labels, and provides examples for annotators. Based on a ten game subset, we found substantial agreement between event annotations of an expert and five novice annotators (mean kappa 0.81). Arrows from one node to another tag causal chains (forward) and references (backward). Causal chains explain that the customer asking for steak *caused* the waitress to bring back a steak from the kitchen. A reference explains that a waitress who asks “How was your lobster?” is *referring* to the previously served lobster.

Annotators do not require specialized skills, aside from English fluency. We hired people from the Philippines, India, Pakistan, and the U.S. via oDesk.com. Seven people completed annotation of 1,000 logs in 415 hours total, for about $3,000. We manually spot checked tags for quality, iterating on corrections with annotators. Work was spread over two months, but seven people working 40 hours/week could complete 415 hours of work in 1.5 weeks.

Once logs have been annotated, we extract all unique utterances included in events, and have humans cluster them semantically by dragging utterances that serve the same purpose into folders. An annotator might drag “Hi” and “Hello” into one folder, and “I’m ready for the bill” and “Check please” into another folder. Prior to clustering, we collapse variables based on a hand-crafted, domain-specific ontology (e.g. “Can I have steak?” and “Can I have salmon?” merge into “Can I have `[FOOD]`?”). Manually grouping utterances can be accomplished with minimal training or specialized knowledge, and allows for flexible, fine-grained groupings. We are clustering in-house, but plan to outsource this in the future.

Automatic Pattern Discovery

We learn a dictionary of discreet sequences, representing events, from the annotations. This process first requires actions and utterances from the text-based logs to be transformed into discreet tokens, stored in an *Action Lexicon* (AL) and *Dialogue Library* (DL).

We generate the AL by recording every unique action in the entire corpus of 10,027 logs. Actions are context-sensitive and role-dependent, stored with pre- and post-conditions based on observed state changes (e.g. `<pickup(waitress, pie), pre: on(counter, pie), post: holding(waitress, pie)>`). Our AL has 10,198 unique actions. We refer to indices into the AL as Action IDs (ACTIDs).

The DL stores unique utterances as sequences of *key words* -- any word observed in at least 25 games. We prune non-keys from the previously clustered utterances, and refer to resulting key word strings as *signatures*. A folder of signatures is a *signature set*, given a unique ID (SSID).

Using the AL and DL, we compile logs into discreet sequences of time-coded ACTID and SSIDs. Fluidly intermixing physical and dialogue actions as a common currency is inspired by the concept of *speech acts* (Austin1962). Time codes associate annotations with tokens in compiled logs, allowing extraction of each unique event pattern. Low-level events are stored in the Event Dictionary (ED) as sequences of ACTIDs and SSIDs. Higher-level events are stored as sequences of event start points. Our 100 game subset includes 135

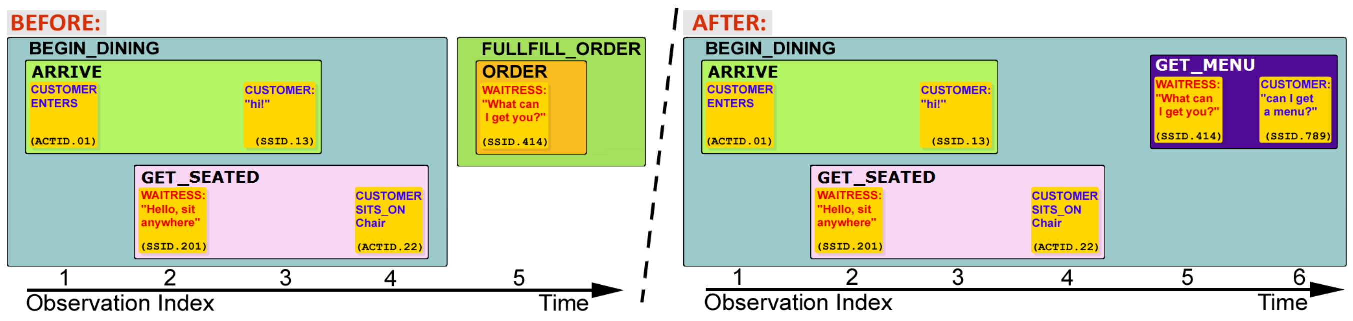


Figure 2: Revising the Plan Recognizer's inferred event hierarchy: before (left) and after (right) observing "can I get a menu?"

patterns for high-level events, and 1,161 for low-level events, composed of 1,051 unique ACTIDs, and 406 SSIDs representing 1,552 unique utterances.

The compiled logs and ED are stored in the Event Log Index (ELI), along with a lookup table indicating the start points of events within log files. The table maps specific event patterns to instances within logs, allowing an NPC to efficiently find logs that match observation sequences at runtime. The ELI also stores associated meta-data, such as references and causal chains.

Data-Driven Episodic Planning

NPCs are driven by an Episodic Planner, which selects actions and utterances through a process that combines plan recognition (Kautz & Allen 1986) with case-based planning (CBP) (Hammond 1990). Cases refer to entire recorded episodes, in the form of annotated logs, indexed by events within. The NPC observes actions and utterances, infers an event hierarchy, proposes games with similar event histories, and critiques proposals until one is found with a valid next action.

For each NPC that exists in the world, an associated agent is running on an AI server (Java), networked with the game engine. As the game engine logs players' actions, resulting state changes, and utterances to a file, the engine broadcasts the same data over the network. The agent uses the AL and DL to process incoming data into discreet *observations* – ACTIDs and SSIDs. Agents process all observations through the same channel, regardless of whether they are associated with another player or the agent itself. Based on these observations, the agent makes decisions about what to do next, which are transmitted to the NPC in the game engine for execution.

The agent tries to understand each new observation within the context of what it has observed previously, using the Plan Recognizer (PR) to infer how new information extends the event hierarchy recognized so far. The PR maintains a hierarchy of token sequences representing events, some of which may be *incomplete*. A sequence is complete if it exactly matches a pattern in the ED. The agent strives to complete incomplete sequences. A new observation may (in order of preference): extend an incomplete sequence, extend a subsequence of an incomplete sequence, start a new sequence, or extend a complete sequence. We refer to each potentially modified sequence as a *candidate*.

The PR iterates over candidates, and enters an exhaustive, recursive process of trying to *apply* and *validate* each. A candidate is applied by truly extending the candidate's corresponding sequence in the hierarchy, or inserting a new sequence into the hierarchy. Possible event labels for a candidate are determined by matching patterns in the ED. Based on the possible labels, the PR tries to use the candidate event to extend, or insert a new, higher-level parent event. Labels are non-committal, and may be disambiguated as new information arrives (figure 2). Each level of the modified hierarchy is validated by matching patterns in the ED. If the entire structure is validated for a candidate, the process is complete -- the observation has been *recognized*, and the modified structure of the hierarchy persists. Otherwise, the modification is reversed, and the process continues until a candidate is validated, or the observation is discarded as unrecognizable.

Once an observation has been recognized, the agent selects the next action by either advancing the current *plan*, or searching for a new *plan*. A plan is a compiled log, and the agent continues following the same plan as long as new observations continue to match the next token in the log, and are validated by the PR. If the observation does not match the next token, or is unrecognizable, the plan is invalidated and the agent re-plans.

Planning begins by iterating over a set of prioritized *interaction goals*. Goals employ a variety of strategies to propose plans that will move the interaction forward coherently. Using the ELI, goals retrieve *proposals* – pointers into logs that begin after a particular sequence of tokens, or at the start of a specified type of event. `G_RespondToSequence` finds logs that contain the most recently extended sequence, and points to the subsequent token. `G_ExtendStructure` finds logs with events that could extend an incomplete higher-level event. `G_CompleteCausalChain` finds logs with events that could complete an initiated, but unresolved, causal chain (e.g. `SERVE_FOOD` or `SERVE_DRINK` if open orders exist). We have implemented eight goals.

For each goal, the agent iterates over the proposals, and re-runs the PR, treating the proposed next action as an imagined observation. Proposals with next actions that cannot be recognized are rejected. Remaining proposals must be validated by a set of *critic* processes. Planning is complete when a proposal is found that is approved by all critics, or when all goals have been evaluated, and no valid proposal has been found. In the case of failure, the agent repeats the process, iterating the focus of attention backward in time to respond to earlier observations.

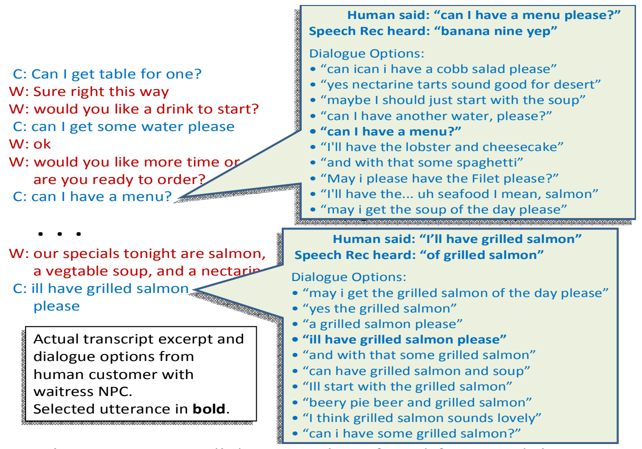
Critic processes ensure future actions maintain coherence, with respect to past observations. We have implemented nine critics. `C_Reference` leverages meta-data to invalidate utterances that refer to events which have not been observed (e.g. do not say "How's your steak?" if steak was never served). `C_ResourceConflict` prevents beginning an event that requires a resource already in use (e.g. the waitress cannot serve beer if her hands are full with steak). `C_InvalidAction` prevents repeatedly trying to execute an action that the game engine reports has failed.

Critics are domain independent, with the exception of `C_Domain`, which can invalidate a proposal based on domain-specific validation functions stored in the Domain Knowledge Manager. Each event type may optionally have procedural preconditions which constrain when that type of event may be initiated or extended. Domain-specific knowledge may be necessary for two reasons: (1) sparse data (e.g. we do not have examples of serving every combination of food, so we encode domain knowledge to ensure we serve entrees before desserts), and (2) discrepancies between NPC and human behavior. Our corpus captures examples of human behavior that we do not want NPCs to execute. Programmatically restricting these behaviors allows them to remain in the corpus for recognition, without risk of execution.

Human Interaction

The interface for human interaction re-uses the previously described planner. When a human controls a player, there is still an associated agent running in the background on the AI server. This agent runs the PR, but does not perform action selection. When the human interacts in the game world, physical actions are broadcast as usual, but human utterances are flagged for further processing by the human's agent, and are ignored by the NPC's agent. Human utterances originate from the text output of the Windows speech recognizer, running with a language model generated from our corpus.

The human's agent is responsible for selecting a list of dialogue options from the corpus, semantically similar to the flagged human utterance. The agent begins by pruning non-key words from the utterance. Next, the agent retrieves a list of SSIDs from the DL for all signature sets that include an utterance containing the key words. The PR generates candidate sequences for all SSIDs that can be recognized as the next action. The agent then iterates over the candidates, applies each, and retrieves proposed plans from the ELI (which point to the SSIDs as the next action).


C: Can I get table for one?
W: Sure right this way
W: would you like a drink to start?
C: can I get some water please
W: ok
W: would you like more time or are you ready to order?
C: can I have a menu?

Human said: "can I have a menu please?"
Speech Rec heard: "banana nine yep"

Dialogue Options:
- "can ican i have a cobb salad please"
- "yes nectarine tarts sound good for desert"
- "maybe I should just start with the soup"
- "can I have another water, please?"
- **"can I have a menu?"**
- "I'll have the lobster and cheesecake"
- "and with that some spaghetti"
- "May i please have the Filet please?"
- "I'll have the... uh seafood I mean, salmon"
- "may i get the soup of the day please"

. . .

W: our specials tonight are salmon, a vegtable soup, and a nectarin
C: ill have grilled salmon please

Human said: "I'll have grilled salmon"
Speech Rec heard: "of grilled salmon"

Dialogue Options:
- "may i get the grilled salmon of the day please"
- "yes the grilled salmon"
- "a grilled salmon please"
- **"ill have grilled salmon please"**
- "and with that some grilled salmon"
- "can have grilled salmon and soup"
- "Ill start with the grilled salmon"
- "beery pie beer and grilled salmon"
- "I think grilled salmon sounds lovely"
- "can i have some grilled salmon?"

Actual transcript excerpt and dialogue options from human customer with waitress NPC. Selected utterance in **bold**.

Figure 3: Top 10 dialogue options found for speech inputs.

Finally, the agent runs the critique process, but rather than stopping at the first approved proposal, the agent continues critiquing, collecting a list of *all* approved proposals.

If this process fails to generate at least five proposals (possibly zero if speech recognition fails), the agent uses context to compensate for failure to understand. The agent falls back to action selection driven by interaction goals, like an NPC, as described in the last section. All proposals from all goals are collected that are not rejected by critics.

The agent now has a list of proposals for utterances deemed valid by the PR and critics. This list is sorted by the count of overlapping words with recognized human input (if any), and the top five utterances are sent to the game engine, for display to the player as dialogue options (figure 3, shows top 10). The human can repeatedly click `MORE` to retrieve the next five options, or `CANCEL` to abort if none of the options are satisfactory. When the human selects a dialogue option for execution, the selected utterance is broadcast as an ordinary unflagged utterance, for processing by agents through ordinary channels.

Evaluation

Our evaluation quantifies how different utterance retrieval methods respond to speech recognition failures and limited coverage. We observed 15 people (with no previous exposure to TRG) playing as a customer, using speech to interact with an NPC waitress.

We divided subjects into three groups of five, each playing under one of three conditions for populating the list of dialogue options: (1) text+context, (2) text-only, (3) context-only. Text+context refers to the system described previously, which selects SSIDs based on key words from speech input, and falls back to interaction goals to compensate for failure to find valid proposals. Text-only presents a sorted list of all utterances in the corpus that match any of the words in the speech input, without using the plan recognizer or critics for filtering. Context-only completely ignores human input, and only relies on the inferred event hierarchy and interaction goals to select the

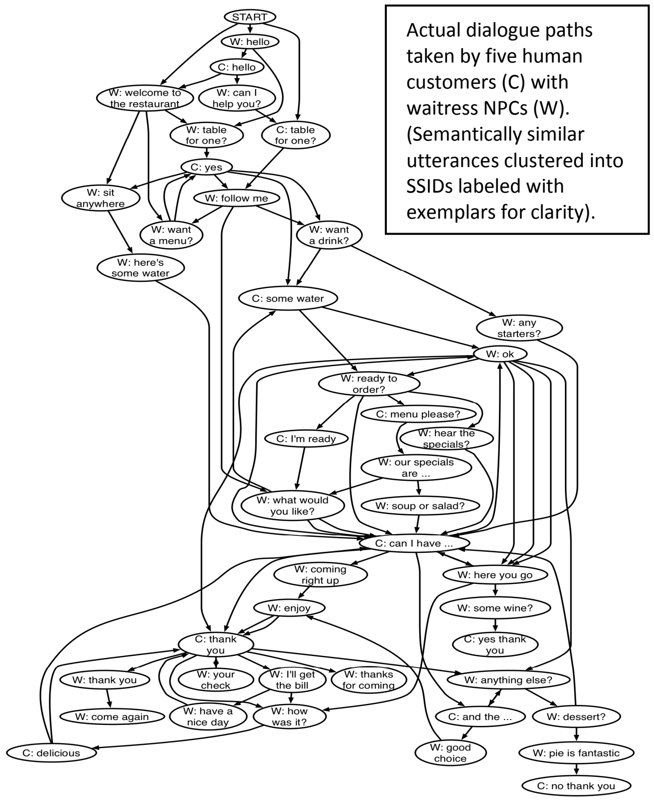


Figure 4: SSIDs observed in 1st 10 inputs of 5 text+context runs.

list of relevant utterances.

Subjects were told to have dinner, and took ~10 minutes to play from entering, through getting seated, having a meal, paying the bill, and departing. Each time the subject spoke to the waitress, s/he was asked to flag as relevant all dialogue options that had the same meaning as what s/he was trying to say. We recorded these flags, the rank in the list of the subject's actual selection, and a count of aborted interactions when no option was selected.

Results and Discussion

The number of speech inputs varies per game. We look at the first 10 in each game, 50 total per condition, for a fair comparison. Table 1 reports that text-only yields the highest percentage of relevant options (total for 50 inputs), and the lowest mean rank of the selected option (closest to the top of the list). However, this is not the whole story. When the speech recognizer fails completely, text-only has no other means of selecting utterances, giving the subject only a failure message, and `CANCEL`. In the text-only condition, subjects aborted 18% more often than text+context (44% vs. 26%, despite similar numbers of recognition failures), due to dissatisfaction with options, or lack of any options. Also, there were two instances where text-only allowed the subject to select an utterance that the plan recognizer could not understand in the current context(due to sparse data), while this never happens in the other conditions where options are filtered by critics.

	text+context	**text-only**	**context-only**
mean selection rank	4.95	**2.11**	6.76
% of opts flagged relevant	38.53	**45.22**	23.48
% of interactions aborted	**26.00**	44.00	32.00
# of plan rec. failures	**0**	2	**0**
# of speech rec. failures	13	10	**5**

Table 1: Comparing 3 methods for populating dialogue options.

Text+context performs better than context-only, validating that the words are important in this scenario, but context can compensate for failure to understand words.

For any speech input, figure 5 plots the likelihood that the subject's selected option will be rank N or less. Text-only delivers the highest likelihood of providing the desired selection at rank five or less, and plateaus shortly thereafter. If the spoken words are recognized correctly, and a similar utterance exists in the corpus, text-only is most likely to provide a desirable option near the top of the list. For each method, the remaining likelihood in the space above the plateau represents the likelihood of aborting. The conditions leveraging context have a higher likelihood of providing a desirable option later in the list, rather than no satisfactory options at all, leading to fewer aborted interactions.

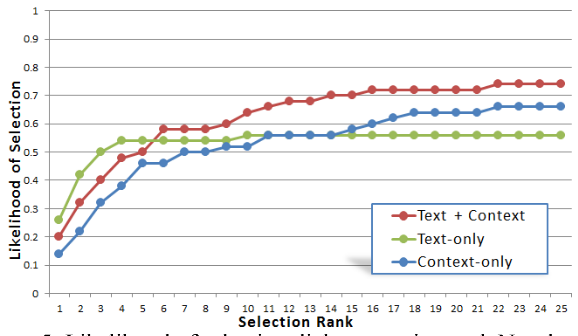


Figure 5: Likelihood of selecting dialogue option rank N or less.

Related Work

Combining crowdsourced content creation, CBP, and speech distinguishes our system from previous interactive narrative systems (Riedl & Young 2003; Cavazza et al. 2002; Magerko 2005). *Façade* accepts typed text input, employing hand-crafted templates to map text to dialogue acts (Mateas & Stern 2004), and compensates for understanding failure with two cleverly designed, self-absorbed NPCs, who can move the narrative forward, ignoring the player when necessary. In an effort to support speech input while playing as the *main character*, who the story cannot progress without, our system proposes contextually appropriate alternatives for unrecognized input by mining data from previous players.

The surprising variability of spontaneous word choice in applications has been documented by Furnas et al. (1987), finding that two people favor the same word < 20% of the time. Inspired by the success of the *How May I Help You* system (Gorin et al. 1997) in coping with varied input by leveraging data from 10,000 customer service calls, TRG was designed to collect examples of restaurant interaction. Other crowdsourcing efforts have collected text-based commonsense data and stories (Singh et al. 2002; Li et al. 2012, Swanson & Gordon 2010), but not at the granularity of actions and utterances required for moment-to-moment interaction with humans.

Hand-crafted representations of situations have enabled inferences required to understand stories (Schank & Abelson 1977); supported plan recognition to improve speech understanding (Gorniak & Roy 2005; Fleischman & Hovy 2006); and powered NPC collaboration and dialogue generation (Hanson & Rich 2010). Learning event hierarchies from annotated logs captures variety and nuance that hand-crafted models are likely to miss.

CBP has been applied to simulation and strategy games (Fasciano 1996; Ortanon et al. 2007). We focus on planning for collaboration with humans. EM-ONE (Singh 2005) employed CBP to model social interaction in a collaborative task. Cases were hand-crafted for one prescribed interaction between NPCs, rather than crowdsourced for interaction with humans.

Corpus-based approaches have been applied to automating chat bots, dialogue generation, and inferring a player's affective state (Huang et al. 2007; Lin & Walker 2011; McQuiggan & Lester 2006). Our work differs in using a data-driven system for both understanding and generation of behavior and dialogue for an embodied NPC, playing a role in a narrative collaboratively with a human.

Conclusion and Future Work

We have evaluated speech interaction with an NPC, and demonstrated that exploiting crowdsourced data and inferred context can compensate for recognition and understanding failures. Future work will focus on scaling up, generalizing, and selecting the single best dialogue option. Migrating all 1,000 annotated logs should decrease text+context's 26% likelihood of aborting, at the risk of introducing new challenges searching and critiquing in real-time. We will evaluate generalization with existing data sets from a virtual sci-fi film set, and a human-robot interaction. Automatic selection of the best option may be able to leverage utterance likelihoods, and data from previous human selections.

Acknowledgements

This research is supported by a grant from the Singapore-MIT GAMBIT Game Lab.

References

Austin, J.L. 1962. *How to Do Things with Words*. Oxford U. Press.

Cavazza, M., et al. 2002. Interacting with Virtual Characters in Interactive Storytelling. In *Proc. of AAMAS.*

Fasciano, M. 1996. Real-time Case-based Reasoning in a Complex World. *Technical Report TR-96-05*. U. of Chicago

Fleischman, M. and Hovy, E. 2006. Taking Advantage of the Situation: Non-Linguistic Context for Natural Language Interfaces to Interactive Virtual Environments. In *Proc. of IUI.*

Furnas, G., et al. 1987. The Vocabulary Problem in Human-System Communications. *Communications of the ACM*, 30.

Gorin, A., et al. 1997. How may I help you? *Speech Communication*, 23(1-2).

Gorniak, P., and Roy, D. 2005. Probabilistic Grounding of Situated Speech using Plan Recognition and Reference Resolution. In *Proc. of ICMI.*

Hammond, K.F. 1990. Case Based Planning: A Framework for Planning from Experience. *Cognitive Science*, 14(3).

Hanson, P. and Rich, C. 2010. A Non-Modal Approach to Integrating Dialogue and Action. In *Proc. of AIIDE.*

Huang, J. et al. 2007. Extracting Chatbot Knowledge from Online Discussion Forums. In *Proc. of IJCAI.*

Kautz, H. and Allen, J. 1986. Generalized Plan Recognition. In *Proc. of Natl. Conf. on AI.*

Li, B. et al. 2012. Learning Sociocultural Knowledge via Crowdsourced Examples. In *Proc. of HCOMP*.

Lin, G. and Walker, M. 2011. All the World's a Stage: Learning Character Models from Film. In *Proc. of AIIDE.*

Magerko, B. 2005. Story Representation and the Interactive Drama. In *Proc. of AIIDE.*

Mateas, M., and Stern, A. 2004. Natural Language Understanding in Façade: Surface-text Processing. In *Proc. of TIDSE.*

McQuiggan, S., and Lester, J. 2006. Learning Empathy: A Data-driven Framework for Modeling Empathetic Companion Agents. In *Proc. of AAMAS.*

Orkin, J. and Roy, D. 2007. The Restaurant Game: Learning social behavior and language from thousands of players online. *Journal of Game Development*, 3(1).

Orkin, J. and Roy, D. 2009. Automatic Learning and Generation of Social Behavior from Collective Human Gameplay. In *Proc. of AAMAS.*

Orkin, J., Smith, T., and Roy, D. 2010. Behavior Compilation for AI in Games. In *Proc. of AIIDE.*

Ortanon, S., et al. 2007. Case-based Planning and Execution for Real-time Strategy Games. In *Proc. of ICCBR.*

Riedl, M. and Young, R.M. 2003. Character-Focused Narrative Planning for Execution in Virtual Worlds. In *Proc. of ICVS.*

Schank, R., and Abelson, R. 1977. *Scripts, Plans, Goals, and Understanding*. Lawrence Erlbaum Associates.

Singh, P., et al. 2002. Open Mind Common Sense: Knowledge Acquisition from the General Public. In *Proc. of ODBASE.*

Singh, P. 2005. *EM-ONE: An Architecture for Reflective Commonsense Thinking*. Ph.D. dissertation. MIT.

Swanson, R., and Gordon, A. 2010. A Data-Driven Case-Based Reasoning Approach to Interactive Storytelling. In *Proc. of ICIDS.*