An Approach to the Automated Evaluation of Pipeline Architectures in Natural Language Dialogue Systems

Eliza Margaretha[*] and **David DeVault**
USC Institute for Creative Technologies, 12015 Waterfront Dr., Playa Vista, CA 90094
elizam@coli.uni-saarland.de
devault@ict.usc.edu

Abstract

We present an approach to performing automated evaluations of pipeline architectures in natural language dialogue systems. Our approach addresses some of the difficulties that arise in such automated evaluations, including the lack of consensus among human annotators about the correct outputs within the processing pipeline, the availability of multiple acceptable system responses to some user utterances, and the complex relationship between system responses and internal processing results. Our approach includes the development of a corpus of richly annotated target dialogues, simulations of the pipeline processing that could occur in these dialogues, and an analysis of how system responses vary based on internal processing results within the pipeline. We illustrate our approach in two implemented virtual human dialogue systems.

1 Introduction

Natural language dialogue systems are typically implemented as complex modular systems, with a range of internal modules performing tasks such as automatic speech recognition (ASR), natural language understanding (NLU), dialogue management (DM), natural language generation (NLG), and speech synthesis (TTS). A common design is for systems to adopt a pipeline architecture. In a pipeline, each user utterance is processed in a series of successive processing steps, with the output of each module serving as the input of the next module, until the system's response is determined.

[*]Now at Saarland University, Germany.

While there are many approaches to dialogue system evaluation (see e.g. (Walker et al., 1997; Eckert et al., 1997; Walker, 2005)), in many ways, the primary data for assessing the performance of a dialogue system comes from the collection of live interactive dialogues between an implemented system and members of its intended user population. Yet, live dialogue-based evaluation suffers from a number of limitations and drawbacks. Each dialogue set can be expensive and time-consuming to collect, and may only reflect a specific version of a system under active development. Additional effort is also generally necessary to identify specific system responses as problematic or unacceptable. Further annotation and analysis is then necessary to diagnose and pinpoint the cause of the problematic responses, so that the relevant pipeline module(s) may be improved.

In this paper, we present and discuss an approach to performing automated evaluations of pipeline architectures. Our approach involves the development of a corpus of annotated *target dialogues,* starting from Wizard-of-Oz data. Our automated evaluation assesses the support for these target dialogues in a pipeline system architecture. It is not designed as a substitute for live system evaluations, but rather as a complement to them which may help to alleviate some of these challenges to understanding system performance and streamlining development. In particular, unlike the PARADISE framework (Walker et al., 1997), which aims to evaluate dialogue agent *strategies* — by relating overall user satisfaction to various other metrics (task success, efficiency measures, and qualitative measures) — our approach takes the agent's dialogue strategy for granted (in

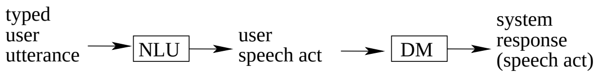

Figure 1: Simplified pipeline architecture.

the form of a set of target dialogues that exemplify the desired strategy), and instead zooms in and aims to directly evaluate the dialogue system's *module pipeline.* Specifically, our approach quantifies the ability of the pipeline to replicate the processing steps needed to reproduce a set of target responses. In our analysis, we place a special emphasis on the possible lack of consensus among human annotators about what the processing results should be. We do not aim to further analyze the system's live dialogue behavior in terms of user satisfaction, task success, or other global measures.

2 Research Setting

The work presented in this paper has been designed to support the dialogue behavior of two virtual human systems, the SimCoach and Tactical Questioning (TACQ) systems. SimCoach (Rizzo et al., 2011) is an on-going project aiming at empowering military personnel and their significant others with online healthcare assistance for Post-Traumatic Stress Disorder (PTSD), depression, and family-related problems. The SimCoach character encourages users to talk about any concerns or problems they may have. TACQ (Gandhe et al., 2008) is designed to support simulation and training for tactical questioning skills, and provides virtual humans who have information but will not answer certain questions unless the user cooperates by agreeing to their requests, offering promises in their favor, and so on. In this work, we have developed target dialogues for the Amani character, who has been an eyewitness of a recent shooting incident.

For simplicity, in the experiments reported in this paper, we have used simplified versions of these two dialogue systems. The simplification removes ASR from TACQ,[1] and removes NLG and TTS from both systems. This yields a simple two-module pipeline architecture that we depict in Figure 1. Note that the input to NLU is a typed English utterance, and the output of the NLU module (also the input to the DM module) is a speech act representation. The output of the DM, which we treat here as the system's response to the user, is also a speech act representation. Both of these systems use statistical classification models for NLU (Leuski and Traum, 2010; Sagae et al., 2009), and finite state machine models for DM (Gandhe et al., 2008; Rizzo et al., 2011).

3 Target Dialogues

Target dialogues are annotated versions of dialogues a system designer would like the system to support.

3.1 Developing Target Dialogues

Wizard-of-Oz (WoZ) and role play dialogues provide valuable data to designers of dialogue systems, especially in the form of natural dialogue data and insights into human-level performance and strategies for the specific dialogue task. However, in practice, system builders may not be able to implement all of the strategies and competences of the wizards or role players, and simplifications may be needed.

SimCoach target dialogues were developed from a collection of 10 WoZ dialogues in which clinicians (wizards) and veterans (users) interacted with each other. We also built Amani target dialogues for TACQ starting from 19 WoZ dialogues. Each user utterance and wizard's response was annotated with a target NLU speech act and one or more target DM speech acts (i.e., the system response).[2] The 10 SimCoach target dialogues contain 376 user utterances and 547 target system response speech acts. The 19 Amani target dialogues contain 317 user utterances and 354 target system response speech acts. For excerpts of the SimCoach and Amani target dialogues, see Tables A.1 and A.2 in the Appendix.

To create our target dialogues, we adjusted the WoZ dialogues to reflect a number of system design limitations as well as wizard deviations from the desired dialogue policy. These changes included removing unsupported wizard utterances and subdialogues, inserting or reordering system responses due to wizard mistakes, and introducing clarification subdialogues for unsupported user utterances.

[1]SimCoach always uses an instant messaging style typed input interface.

[2]For both SimCoach and TACQ, the DM may generate one or multiple speech acts in response to a user utterance.

3.2 Formalizing Target Dialogues

Let $P = \langle p_1, ..., p_k \rangle$ be the pipeline in a system containing k modules. We use S_t to denote the *pipeline state*, which includes the internal states of any modules that maintain an internal state, at time t.

For a user input x_t that occurs at time t, when the pipeline state is S_t, we write $A(P, S_t, x_t) = \langle y_1, ..., y_k \rangle$ to represent the actual sequence of outputs from the pipeline modules, where y_i is the output of module p_i for $i = 1...k$.

For a variety of reasons, these actual module outputs may differ from the target module outputs for this input and pipeline state. Let $T(P, S_t, x_t) = \langle z_1, ..., z_k \rangle$ be the *target pipeline response* to input x_t, i.e. the sequence of target outputs from each of the pipeline modules.

A target dialogue $\mathcal{D} = \langle (x_1, T_1), ..., (x_N, T_N) \rangle$, then, is a sequence of user inputs and corresponding target pipeline responses. Specifically, for time $t = 1...N$, $T_t = T(P, S_t^*, x_t) = \langle z_1, ..., z_k \rangle$ is the target pipeline response to input x_t, where S_t^* is the *target pipeline state* at each time t.

An important detail is that the *target pipeline state* S_t^* is the state that the pipeline *would* be in if all previous user inputs had triggered exactly the target pipeline responses. Formally, let S_1^* be the initial state of the dialogue system pipeline. Then, let $S_{t+1}^* = \mathrm{update}(S_t^*, x_t, T_t)$, where we use an update function to capture the effect on the internal state of the pipeline of the target response T_t to x_t. Note that the target pipeline state may differ from the actual pipeline state, if an actual pipeline response differs from the target pipeline response. For example, if a previous user utterance was misunderstood by an NLU module, then at run-time, the actual information state inside the DM module would reflect this earlier misunderstanding, while the target pipeline state would include a corrected version of the information state. Using corrected information states, and corrected pipeline states more generally, enables the utterances within a target dialogue to be considered independently in a pipeline evaluation.[3]

We can say that a pipeline P is *compatible* with a target dialogue $\mathcal{D} = \langle (x_1, T_1), ..., (x_N, T_N) \rangle$ iff $A(P, S_t^*, x_t)[k] = T_t[k]$ for all $t = 1...N$. In other words, for every user utterance, the actual system response, as emitted by the last (k^{th}) module in the pipeline, matches the target system response.[4] Both the SimCoach and TACQ pipelines are *compatible* in this sense with their target dialogues (Section 3.1).

User Utterance	NLU Speech Act	DM Response
Having difficulty sleeping... bad dreams.. Wake up a few times every night	answer.observable. sleeping-problems	question. depression-pre-check-list.1
	answer.observable. wakeup-generic	question. depression-pre-check-list.1
	answer.observable. wakeup-nightmare	question. ptsd-pre-checklist.1

Table 1: Sample of Different NLU Speech Acts

3.2.1 Addressing the Lack of Consensus

A considerable challenge in the improvement of pipeline performance is the *lack of consensus* about the desired internal processing steps: different system designers or human annotators often disagree about what the intermediate results should be. For example, in a system such as TACQ or SimCoach, there may be substantial disagreement among human annotators about the correct NLU output for each utterance; see e.g. (Artstein et al., 2009). Table 1 exemplifies 3 different possible NLU speech act annotations for a user utterance to SimCoach. Note that for the first two, the DM outputs the same system response (which incidentally is the target response). However, the third speech act yields a different response. In our automated evaluations, rather than trying to resolve all disagreements, our approach is to characterize the frequency with which these kinds of phenomena occur in the pipeline.

To support this analysis, for a target dialogue $\mathcal{D} = \langle (x_1, T_1), ..., (x_N, T_N) \rangle$, we assume then that each input x_t is associated not only with the target pipeline response T_t, but also with a collection of annotations $A_t = \langle a_1, ..., a_k \rangle$. These annotations may be derived from a number of independent sources

[3]It also highlights how our pipeline evaluation results do not translate directly into performance metrics for live dialogues, as deviations and errors in system responses in live dialogues may affect the subsequent interaction in ways that are difficult to predict and deviate substantially from the target dialogues.

[4]A technical detail: for both SimCoach and TACQ, the DM sometimes emits multiple speech acts; to accommodate these cases, for now we treat the target DM output as a set of speech acts $\mathcal{A}$, and count each actual output DM speech act as an independent match if it matches *any* speech act in $\mathcal{A}$ (ignoring order). A more complex matching scheme could be employed.

$\mathcal{S} = \{s_1, ..., s_l\}$, and we write $a_i(s) = w_i$ to denote the correct output w_i for module p_i according to annotation source $s \in \mathcal{S}$. These independent "annotation sources" might be human annotators, or competing module algorithms, for example.

We can then capture the hypothetical effect of using annotation source s in place of some module p_i within the pipeline. To do so, we consider the effect of replacing the output of module p_i with $a_i(s)$, and using this as the input to subsequent modules in the pipeline. Let $P_{i+1}^k = \langle p_{i+1}, ..., p_k \rangle$ be the remainder of the pipeline, starting at module p_{i+1}. For input x_t, we can notate the *hypothetical pipeline response*, if module i were replaced by annotation source s, by $H(P_{i+1}^k, S_t^*, a_i(s)) = \langle y_{i+1}, ..., y_k \rangle$. We will write $h_t^{s\backslash i}$ for the hypothetical system response to the user input at time t, if source s were substituted for the output of module i: $h_t^{s\backslash i} = H(P_{i+1}^k, S_t^*, a_i(s))[k] = y_k$. For a target dialogue of length N, we can summarize the frequency with which the hypothetical pipeline response would match the target system response by a performance measure:

$$\mathcal{P}_{\text{strict}} = \frac{1}{N} \sum_{t=1}^{N} \text{match}(h_t^{s\backslash i}, T_t[k])$$

where $\text{match}(x, y) = 1$ if $x = y$ and 0 otherwise.[5]

A second form of lack of consensus issue is the existence of *multiple acceptable system responses* within a system. Returning to the example in Table 1, system designers might decide that either of the two system responses here would be acceptable. In some cases, actual NLU outputs which differ from the target NLU output will simply result in the system giving alternative acceptable system responses, as in this example. In other cases, they may lead to unacceptable system responses.

We measure the frequency with which these phenomena occur as follows. For a target dialogue $\mathcal{D} = \langle (x_1, T_1), ..., (x_N, T_N) \rangle$, let each input x_t be associated with a set $R_t = \{r_1, ..., r_m\}$ of system responses which differ from the target system response $T_t[k]$, but are also acceptable in design terms. Given these alternative responses, we can then define a more permissive performance measure:

$$\mathcal{P}_{\text{multiple}} = \frac{1}{N} \sum_{t=1}^{N} \text{match}(h_t^{s\backslash i}, T_t[k], R_t)$$

[5]This strict agreement measure can be easily generalized to measure the proportion of matches in a set of target dialogues.

NLU speech act source	Percent of NLU speech acts identical to... (N=317)		Percent of system response speech acts identical to... (N=354)	
	the target NLU speech act (target)	the target or other acceptable NLU speech act (human$_{\text{all}}$)	a target system response speech act	a target or acceptable system response speech act
target	100%	100%	99.4%	100%
human$_1$	79.3%	95.4%	84.2%	88.4%
human$_2$	76.7%	99.7%	86.7%	93.8%
human$_3$	59.3%	90.2%	69.6%	78.8%
NPCEditor	42.3%	50.5%	55.3%	57.4%

Table 2: TACQ Amani Evaluation Results

where

$$\text{match}(h_t^{s\backslash i}, T_t[k], R_t) = \begin{cases} 1 & \text{if } h_t^{s\backslash i} = T_t[k] \\ 1 & \text{if } h_t^{s\backslash i} \in R_t \\ 0 & \text{otherwise} \end{cases} .$$

4 Results

4.1 Annotations and Results for TACQ

We collected a range of annotations for the 19 TACQ Amani target dialogues, including 6 sources of NLU speech acts for the 317 user utterances: target (the target NLU speech act for each utterance); 3 independent human annotations of the best NLU speech act for each utterance; human$_{\text{all}}$ (a set containing *all* of the alternative acceptable NLU speech acts for each utterance, according to the same single researcher who prepared target); and NPCEditor, the NLU speech act output from NPCEditor (Leuski and Traum, 2010), the NLU module for TACQ.

We analyzed the effect of differing NLU speech act sources on the responses given by the system. We present the results in Table 2. (For a detailed processing example, see Table A.2 in the Appendix.) The first (leftmost) column of numbers shows the percentage of NLU speech acts from each source that are identical to the target NLU speech act. These results highlight how human annotators do not always agree with each other, or with the target. The agreement among the human annotators themselves, measured by Krippendorf's alpha (Krippendorff, 2007) is 0.599 (see also (Artstein et al., 2009)). In the second column of numbers, we tabulate the frequency with which the NLU speech acts are present in human$_{\text{all}}$. While these numbers are higher, they do not reach 100% for the human annotators, suggesting that a single annotator is unlikely to be able to circumscribe all the NLU speech acts that other annotators might find acceptable.

Despite the frequent disagreements among human annotators, this evaluation shows that the impact on the target system responses is less than might be expected. In the third column of numbers, we calculate $\mathcal{P}_{\text{strict}}$ which measures the effect of using each of NLU sources, in place of the NLU module's actual output, on the pipeline's ability to produce the target response. As the table implies, the pipeline often produces the target system response (third column) even when the NLU source disagrees with the target (first column). Indeed, for all the NLU sources except for target, the pipeline is significantly more likely to produce the target system response than the NLU source is to produce the target NLU speech act (Wilcoxon test, $p < 0.001$ for each source).

We also calculate $\mathcal{P}_{\text{multiple}}$ (last column) which measures the effect of using each NLU source on the pipeline's ability to produce either the target or any other acceptable system response. As the table shows, the actual system responses are often acceptable when they differ from the target responses. Although this effect seems weaker for NPCEditor, Wilcoxon tests reveal that for every source other than target, the differences between $\mathcal{P}_{\text{strict}}$ and $\mathcal{P}_{\text{multiple}}$ are significant at $p < 0.005$. This evaluation confirms that the pipeline is significantly more likely to deliver an acceptable system response than a target response, and helps quantify to what extent NLU outputs that differ from the target remain problematic for the pipeline performance.

4.2 Annotations and Results for SimCoach

We gathered a set of annotations for the 10 SimCoach target dialogues, including 3 sources of NLU speech acts for the 376 user utterances: target, human_1, and mxNLU (the NLU speech act output from mxNLU (Sagae et al., 2009), the NLU module for SimCoach). We present the evaluation results in Table 3. As the table shows, our independent human annotator often disagreed with the target NLU speech act. Despite the 72.1% agreement rate, the system's response to the human NLU speech act agreed with the target response 93.3% of the time.

In comparison, mxNLU shows somewhat higher

NLU speech act source	NLU speech acts identical to target (N = 376)	System response speech acts identical to target (N = 547)
target	100%	100%
human_1	72.1%	93.3%
mxNLU	75.3%	91.1%

Table 3: SimCoach Evaluation Results

agreement (75.3%) with the target NLU annotation. While this might at first suggest "super-human" NLU performance, in reality it is because the target NLU annotation was constructed in very close consultation with the training data for mxNLU.[6] Despite showing higher agreement with target NLU speech acts, the system responses were not more likely to match the target system responses with mxNLU. The explanation is that disagreements for mxNLU were more serious, reflecting more misunderstandings and failures to understand than occur with a human annotator, and more deviations from the target responses. This highlights the value of looking beyond the performance of individual modules.

5 Conclusions and Future Work

We have presented an approach to performing automated evaluations of pipeline architectures, and demonstrated its application in two implemented virtual human dialogue systems. The pipeline evaluation provided several insights into the current pipeline performance, including what performance would be attainable if human-level NLU were possible. In future work, we would like to expand beyond our simplified two-module pipeline, and investigate the connection between our automated pipeline evaluations and performance in live dialogues.

Acknowledgments

We thank our reviewers, Sudeep Gandhe, Fabrizio Morbini, and David Traum. The project effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

[6]The exact target dialogue utterances were not in the mxNLU training data, but similar utterances were inspected in constructing the target dialogues.

References

R. Artstein, S. Gandhe, M. Rushforth, and D. Traum. 2009. Viability of a simple dialogue act scheme for a tactical questioning dialogue system. In SemDial Workshop, pages 43–50.

W. Eckert, E. Levin, and R. Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *Proc. IEEE ASR Workshop*, pages 80–87.

S. Gandhe, D. DeVault, A. Roque, B. Martinovski, R. Artstein, A. Leuski, and et al. 2008. From domain specification to virtual humans: An integrated approach to authoring tactical questioning characters. Proceedings of Interspeech-08.

K Krippendorff. 2007. Computing krippendorff's alpha-reliability, June.

A. Leuski and D. Traum. 2010. NPCEditor: A tool for building question-answering characters. In LREC.

A. Rizzo, B. Lange, J.G. Buckwalter, E. Forbell, J. Kim, K. Sagae, J. Williams, B.O. Rothbaum, J. Difede, G. Reger, T. Parsons, and P. Kenny. 2011. An intelligent virtual human system for providing healthcare information and support. In et al. Westwood, J.D., editor, *Technology and Informatics*. IOS Press.

K. Sagae, G. Christian, D. DeVault, and D. R. Traum. 2009. Towards natural language understanding of partial speech recognition results in dialogue systems. In Short Paper Proceedings of NAACL HLT.

M. A. Walker, D. J. Litman, C. A. Kamm, A. A. Kamm, and A. Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. pages 271–280.

M. A. Walker. 2005. Can we talk? methods for evaluation and training of spoken dialogue systems. *Language Resources and Evaluation*, 39(1):pp. 65–75.

Appendix

t	User Utterance (x_t)	Target NLU Speech Act (t_1)	Target System Response (t_2)	Textual Version of Target System Response
9	my husband seems distant, and we have been arguing a lot more lately	answer.observable. family-problem	question.bio-info. has-kids	Does he have children?
10	yes, 2	answer.yes	question.family-pre-checklist.6	In his family, do people collaborate together to find the best way to solve problems?

Table A.1: Excerpt from a SimCoach Target Dialogue.

t	User Utterance (x_t)	Source of NLU Speech Act	NLU Speech Act (gloss)	System Response Speech Acts (gloss)
1	hi amani.	NPCEditor	*hello*	*hello*
		target NLU	*hello*	*hello*
2	i wanted to talk to you about the recent shooting that occurred	NPC Editor	*Tell me more about the_ incident*	*location of the_incident is the_shop*
		target NLU	*Is amani willing to talk?*	*amani is willing to talk*
3	do you know who was responsible?	NPC Editor	*What is perpetrator of the_ incident ?*	*perpetrator of the_incident is Saif*
		target NLU	*What is name of strange_man ?*	*player should offer 'give-safety'*

Table A.2: Excerpt from a TACQ target dialogue, including pipeline module processing.