Investigation of Error Simulation Techniques for Learning Dialog Policies for Conversational Error Recovery

Maryam Fazel-Zarandi*
Amazon Inc, USA
fazelzar@amazon.com

Longshaokan Wang*
Amazon Inc, USA
longsha@amazon.com

Aditya Tiwari
Amazon Inc, USA
aditiwar@amazon.com

Spyros Matsoukas
Amazon Inc, USA
matsouka@amazon.com

Abstract

Training dialog policies for speech-based virtual assistants requires a plethora of conversational data. The data collection phase is often expensive and time consuming due to human involvement. To address this issue, a common solution is to build user simulators for data generation. For the successful deployment of the trained policies into real world domains, it is vital that the user simulator mimics realistic conditions. In particular, speech-based assistants are heavily affected by automatic speech recognition and language understanding errors, hence the user simulator should be able to simulate similar errors. In this paper, we review the existing error simulation methods that induce errors at audio, phoneme, text, or semantic level; and conduct detailed comparisons between the audio-level and text-level methods. In the process, we improve the existing text-level method by introducing confidence score prediction and out-of-vocabulary word mapping. We also explore the impact of audio-level and text-level methods on learning a simple clarification dialog policy to recover from errors to provide insight on future improvement for both approaches.

1 Introduction

Spoken dialog agents are becoming a prevalent tool in personal assistant devices. Such human-like interfaces create a rich experience for users by enabling them to complete many tasks hands-free and eyes-free in a conversational manner. Depending on how complicated a task is, multiple rounds of conversation may be needed for the assistant to fully understand user requests. To achieve this, these interfaces require the design of complex dialog policies which can generate appropriate responses to queries and steer the conversation. Designing dialog policies is difficult and requires a plethora of data, because responding to a user is conditioned on the task, the user goals and preferences, and the dialog history. Additionally, environmental noise and ambiguous user utterances add to the complexity. Furthermore, collecting data is expensive and time consuming due to human involvement. It may also be difficult to get engagement from real users in the early stages of the dialog policy development. Consequently, it is necessary to leverage semi-supervised or unsupervised methods which require less human labeling and can account for many branching user interactions. Reinforcement learning (RL) has become one such method for training dialog agents (e.g., [29, 37, 15]). However, RL training may require the use of a user simulator which can account for realistic branching user paths [25].

*Equal contributions.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Ensuring a successful transfer of an RL policy from simulation to production requires a moderately accurate simulation [31, 24, 3]. To improve the realism of RL policies, current methods in user simulators induce errors at different levels to mimic noisy communication channels. More specifically, existing methods induce errors at audio [8], phoneme [30, 35], text [21, 26, 28] or semantic [23, 32, 19] level. In this paper, we compare different error simulation methods and investigate which method simulates the most realistic errors and is thus most suitable for RL-like tasks; we also improve the existing methods in the process. The audio and text level approaches are the easiest to operationalize and generalize to new domains (see Section 2), and hence will be our focus.

The contributions of this paper are as follows: First, we conduct a comparative study among different error simulation methods. In particular, we evaluate audio- and text-level methods in terms of word error distribution, automatic speech recognition (ASR) confidence score distribution, whether the produced errors are distinguishable from real errors, and their effect on natural language understanding. We show that the text-level method is comparable to the audio-level method in the above metrics, and should be the preferred method for its speed and simplicity. Second, we improve the existing text-level method by introducing ASR confidence score prediction and out-of-vocabulary word mapping. Confidence scores are important signals for error recovery policies. To this end, we present a simple method to predict ASR confidence scores from the simulated ASR output. To our knowledge we are the first to tackle the problem of score prediction from the simulated ASR output in the absence of audio signals. Third, we improve the discriminator approach used by [35] for evaluating error simulation, by including predicted ASR score and word error rate (WER) features as part of the input. Lastly, we explore the impact of audio- and text-level methods on learning a simple clarification policy by evaluating them on held-out annotated data and conducting a user study.

2 Background

When a user talks with a conversational agent, the audio signal is first translated into text using an ASR module. Using a natural language understanding (NLU) module, the agent extracts meaning from a possibly noisy input, usually in terms of user intents and slots, where an intent indicates the user's intention (e.g., getting the plot) and slots represent information about a particular entity (e.g., movie title). Finally, the agent decides what to respond to the user based on its dialog policy. To be successful, the dialog policy should allow the agent to recover from conversational errors such as ASR and NLU mis-recognition.

A dialog system can be formalized as a Markov Decision Process (MDP) [16]. An MDP is a tuple $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma>$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability function, $\mathcal{R}$ is the reward function, and γ is the discount factor. In the RL framework, at each time step *t*, the agent observes state $s_t \in \mathcal{S}$ and selects action $a_t \in \mathcal{A}$ according to its policy ($\pi : \mathcal{S} \rightarrow \mathcal{A}$). After performing the selected action, the agent receives the next state s_{t+1} and a scalar reward r_t. The trajectory restarts after the agent reaches a terminal state. RL solvers have been used to find the optimal dialog policy (e.g., [29, 37, 15]). In this context, at each turn the agent acts based on its understanding of what the user said, and reward function is modeled in terms of various dimensions such as per-interaction user satisfaction, accomplishment of the task, efficiency of interaction, and dialog duration. Recently, deep RL has also been applied to the problem of dialog management and has shown improvements over rule-based systems (e.g., [5, 38, 7]).

An important challenge in using RL for learning dialog policies is creating realistic user simulators that can generate natural conversations similar to a human user [25]. In previous works, researchers have focused on the development of different types of user simulators (e.g., [6, 27, 4, 11, 25, 1, 17]). However, existing systems are limited with respect to accurately simulating errors that a dialog agent might face [18, 8, 35].

Errors can be simulated at audio, phoneme, text or semantic level. At audio level [8], audio signals are synthesized from reference texts, noise is injected into the audio signals, and then noisy texts are obtained from decoding the noisy audio signals. This approach does not require training data, can leverage the text-to-speech and ASR decoding modules readily available in dialog systems, and can directly provide ASR confidence scores as part of the output. However, it is computationally expensive, and mimicking real life background noise can be difficult due to its diversity. Additionally, the ASR WER on synthesized speech is typically lower than on real speech. At phoneme level [30, 35], reference texts are converted to phonemes, phoneme replacements are sampled from a

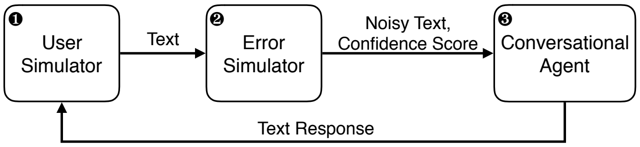


Figure 1: User simulator and conversational agent interaction. We use the text response generated by the agent before sending it to TTS to eliminate the need for ASR/NLU on the user simulator side.

phoneme confusion matrix, and then noisy phonemes are converted to noisy texts. This approach relies on high quality conversion modules between texts and phonemes, requires predicting the ASR confidence scores, and if a generic phoneme confusion matrix is used, the simulated errors might differ from those in a particular domain. At text level [21, 26], n-gram confusion matrix is constructed from transcribed training data, reference texts are probabilistically partitioned into n-grams, then n-gram replacements are sampled from the confusion matrix. This approach can directly match the distribution of error types in its output with the one in the training data, does not require additional modules, and is computationally efficient, but requires transcribed training data, predicting the ASR confidence scores, and handling of out-of-vocabulary words (we later present approaches to overcome the last two shortcomings). At semantic level, replacements of intents and slots are created by either sampling from a semantic confusion matrix [23], running multiple classifiers [32], or sampling uniformly [19]. This approach skips any conversion or decoding and is thus the fastest approach, but it often requires unrealistic simplifying assumptions such as the independence between intent and slot [32], requires training data with semantic annotations, and is difficult to generalize to new domains.

Considering generalizability and the ease of operationalization, we focus on audio- and text-level approaches in our comparative study. We provide more comprehensive comparisons beyond the previous comparative study of [35], which only considers the fluency of generated alternatives and the distributional overlap between the simulated and real output on the training set.

3 Error Simulation

Figure 1 illustrates the interaction between a user simulator and a conversational agent. The user simulator learns to behave like a human to converse with the agent based on the dialog context and generates noisy text using an error simulator. In this section we describe the audio-level and text-level error simulation techniques that we compared in this study. For text-level error simulation, we discuss our improvements to the existing approach.

3.1 Audio-Level Error Simulation

In audio-level error simulation, we take the approach of generating a spoken audio signal from text, adding noise into the audio and then passing the audio through an ASR model to get a hypothesis and confidence score. This process simulates the real-world sources of error such as background noise in a person's home. Concretely, we synthesize an audio signal with the text-to-speech (TTS) service provided by Amazon Polly[2] and contaminate the clean TTS output to achieve a desired signal-to-noise ratio (SNR). Contamination is performed by injecting real noise from randomly selected conversations annotated as background noise to the TTS output. For the purposes of this study, we use different TTS voices and dialects provided by Amazon Polly and select the SNR value such that we match the ASR WER of our corpus. Finally, we pass the noisy audio through an ASR decoder to get the corresponding text and associated confidence score.

3.2 Text-Level Error Simulation

In text-level error simulation, we start with Schatzmann *et al.*'s [26] n-gram confusion matrix approach (see Appendix A for details), which showed marginally better performance than the other error simulation approaches in Wang *et al.*'s (2018) comparative study. To learn realistic dialog policies, we need to predict the ASR confidence score that a dialog system would produce for a simulated ASR output. Schatzmann *et al.* [26] and Thomson *et al.* [32] have mentioned how to assign

[2] `https://aws.amazon.com/polly`

confidence scores to simulated ASR output, but their approaches simply sample from the training set distributions of ASR scores based on whether the simulated ASR output matches the reference without trying to predict the score based on the exact output. Also, to simulate errors on new data, we need to handle words outside the vocabulary of the n-gram confusion matrix constructed from training data. Hence, we improve upon the existing techniques by adding ASR confidence score prediction and out-of-vocabulary word mapping. Note that these improvements are not restricted to one particular technique like Schatzmann *et al.* [26]. In fact, the ASR confidence score prediction from ASR output is also necessary for any phoneme-level error simulation.

Given transcribed training data with reference utterances, ASR hypotheses, and ASR confidence scores, we first construct the n-gram confusion matrix between ASR hypotheses and reference utterances after aligning each hypothesis with its reference by Levenshtein distance. To train the ASR score prediction model, we extract the Term-Frequency-Inverse-Document-Frequency (TFIDF) [20] vectors for hypotheses and references separately, and compute WER related features including WER, sentence length, number of correct words, insertions, deletions, and substitutions for each hypothesis-reference pair. We then concatenate the TFIDF vectors and WER related features to form the input feature matrix. We can formulate the problem as either regression or classification on top of the feature matrix with the ground-truth ASR confidence scores as the targets. One problem with regression is that the predictions cluster towards the mean score due to minimizing mean squared error, creating a clear difference between the distributions of predicted scores and actual scores. In the case of classification, we can first classify the score bin (e.g., bin 1: $[0, 0.1)$, bin 2: $[0.1, 0.2)$), then sample from the training set distribution within the predicted bin to obtain the predicted score.

On test/new reference utterances, we probabilistically partition each utterance into a list of n-grams based on their frequencies in the training data (see [26] for details), sample each n-gram's replacement (including itself which means no error) from the confusion matrix, and predict the ASR confidence score for each reference and simulated hypothesis pair with the trained ASR score model and the same feature extraction. Words not seen during training are mapped to their closest in-vocabulary counterparts through fuzzy-matching[3], but with some probability of staying the same to avoid inflating the substitution errors: Let the out-of-vocabulary word be O, its closest in-vocabulary match be C, and the WER in training data be $\widehat{P}$, then with probability $1 - \widehat{P}$ we let O stay the same and otherwise sample its replacement from the confusion vector for C.

By design, the procedure produces WER and error type (insertion, deletion, substitution) distributions on test set similar to those of training set. To artificially increase or decrease WER, we can adjust the frequencies that each n-gram is substituted with itself (no error) in the confusion matrix.

4 Experimental Results

To compare different error simulation techniques, we used a corpus collected from MovieBot[4], an Alexa Skill [13] that converses with users about existing, new, and forthcoming movies [8]. We used a 75/25 training/test split of the corpus which consists of $64,830$ transcribed user turns.

4.1 Analysis of Simulated ASR Hypotheses

Table 1 reports the relative WER change with respect to the real ASR output, as well as error type distributions for the real ASR output and the output of audio- and text-level error simulation for the

[3] `https://docs.python.org/3/library/difflib.html`

[4] `https://www.amazon.com/Amazon-MovieBot/dp/B01MRKGF5W`

Table 1: Error distribution of real and simulated output on the test data

	Relative WER Change	Substitutions	Insertions	Deletions
ASR Output	0.00%	48.32%	18.67%	33.01%
Audio-Level	$+9.49\%$	36.51%	5.00%	58.49%
Text-Level	-10.40%	52.86%	16.29%	30.86%

Table 2: Evaluation of ASR confidence score prediction models

	Linear Correlation	**Mean Abs. Error**
Baseline	0.377	0.241
Regression	0.777	0.132
Classification	0.736	0.150

test data. In this table, we see that both methods match the WER of the real ASR output (10% relative change is small). The distribution of errors in the text-level approach, however, better matches those of the real ASR output. For the audio-level method the errors are different between real and simulated outputs; whereas the majority of errors in real ASR are substitutions (48%), the majority of errors in simulated ASR are deletions (58%). This is partly due to the fact that WER in real data can be a result of factors other than background noise, such as accents; hence matching the WER in simulated data only through audio noise sometimes leads to too much noise and empty ASR output.

4.2 ASR Confidence Scores for Simulated ASR Hypotheses

We experimented with both classification and regression models for ASR score prediction for text-level error simulation, and evaluated it by comparing its predictions on the real ASR hypotheses with the ground truth ASR scores in the test set. We used Schatzmann *et al.*'s [26] approach as baseline, which samples from the training set score distribution based on whether the hypothesis matches the reference. After preliminary experimentation we chose gradient boosting [10] for both classification and regression. It is possible to explore deep learning models for score prediction, but for relatively simple tasks, TFIDF with an off-the-shelf classifier can produce competitive results [33, 36].

Table 2 shows the linear correlation and Mean Absolute Errors against the real ASR scores for each model. Both regression and classification approaches substantially outperformed the baseline and yielded predictions that are close to the real scores in terms of correlation and MAE. The predictions from the classification approach match the ground-truth scores much better in distribution but have slightly smaller correlation and larger MAE compared to the predictions from the regression approach. It is worth pointing out that regardless of the score model, there is an inherent gap between scores predicted from the hypothesis-reference pairs and real scores, because for the same hypothesis-reference pair, the real score could still vary based on the audio quality.

Figure 2 illustrates the relative changes in the simulated ASR scores compared to the real scores on the test set. We see that the text-level scores with regression are skewed towards the center as a result of minimizing mean squared error. The distributions for the audio- and text-level scores with classification are both close to that of the real scores except for rare spikes, with a Kullback-Leibler divergence of 0.08 and 0.03, respectively. For the audio-based method, the spike in the 0.0-0.1 bin is due to cases where too much noise is added and as a result the ASR model considers the input as noise; the spike in the 0.9-1.0 bin, on the other hand, is due to the characteristics of the TTS output where in certain cases the words are clearly spoken and the added noise does not distort the signal. For text-level scores with classification, the spikes can be explained by the model always predicting

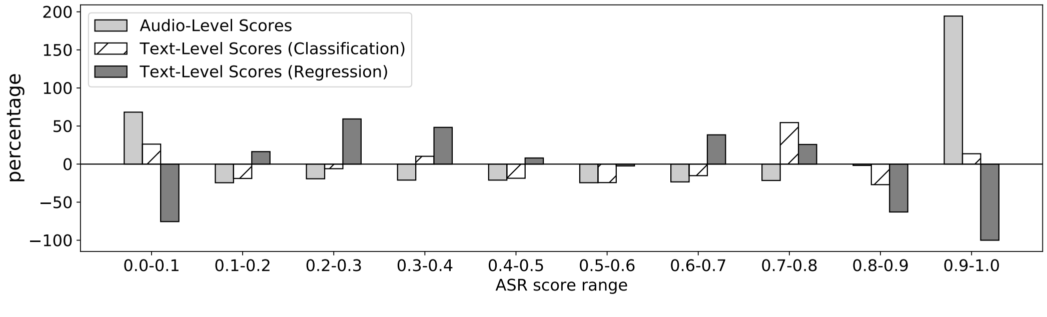


Figure 2: Relative changes in the simulated ASR scores compared to the real scores on the test set

Table 3: Discriminator performance (poorer performance means more realistically simulated output)

	Accuracy	Precision	Recall	F-Score
Audio-Level, Without Score	62.59%	0.688	0.462	0.552
Text-Level, Without Score	60.36%	0.633	0.492	0.554
Audio-Level, With Score	69.01%	0.725	0.614	0.665
Text-Level, With Score (Classification)	75.18%	0.727	0.807	0.765
Text-Level, With Score (Regression)	88.30%	0.855	0.923	0.888
Audio-Level, Without Score, No Duplicates	63.03%	0.619	0.679	0.647
Text-Level, Without Score, No Duplicates	64.95%	0.642	0.676	0.659
Audio-Level, With Score, No Duplicates	65.54%	0.665	0.628	0.646
Text-Level, With Score (Classification), No Duplicates	67.05%	0.665	0.689	0.676
Text-Level, With Score (Regression), No Duplicates	78.33%	0.752	0.846	0.796

the most frequently seen bin for hypothesis-reference pairs that occur multiple times. When we examined the samples that are misclassified to the 0.7-0.8 and 0.0-0.1 bins in the training set, we found that 96% are hypothesis-reference pairs with multiple duplicates that are most frequently in the respective bins but also spread across other score bins.

4.3 Discriminator

To evaluate how realistic the simulated ASR output is for each error simulation method, we adopted Wang *et al.*'s [35] approach of training a discriminator to see how well it can distinguish the simulated ASR output from the real one. We are interested in seeing how realistic the simulated ASR hypotheses are, both on their own and together with the (predicted) ASR scores, so we ran experiments with and without ASR scores in the input. Wang *et al.*'s [35] discriminator takes the ASR hypotheses as input, but we postulate that to better distinguish whether a hypothesis is simulated, the discriminator should compare it with the reference (which was confirmed in our preliminary experiments where adding the reference improved the accuracy by ~2% for the cases considered), hence we included references in the input. We extracted TFIDF vectors from both hypotheses and references, computed WER related features, and optionally concatenated them with ASR scores to form the input matrix, then we fitted a Gradient Boosting Classification model on top to predict whether each hypothesis is simulated or not.

Table 3 shows the test set performance of the discriminator in terms of accuracy, precision, recall, and F-score for different combinations of error simulation methods, input features, and data preprocessing. Poorer discriminator performance means more realistically simulated ASR output. Disregarding ASR scores, the discriminator has low accuracy on both text- and audio-level methods at 60% and 63%, respectively, indicating that the simulated hypotheses from both methods are realistic. Including ASR scores as a predictor, the accuracy for audio-level method improves to 69% despite the fact that its ASR scores are obtained through ASR decoding, but the increase can be likely explained by the score distribution difference seen in Figure 2. In comparison, the accuracy for both text-level methods have much larger increases. For text-level method with regression, it is easy to see why the scores can be a strong predictor, given the large score distribution differences between the real and predicted scores by this method (Figure 2). For text-level method with classification, the reason is more subtle and a result of the score model only predicting the most frequent bin for hypothesis-reference pairs that have multiple duplicates with different real ASR scores. For a more details see Appendix B.

To evaluate how realistic the score predictions are without the superficial influence of duplicate hypothesis-reference pairs, we repeated the experiment after removing such pairs in real and simulated data (last five rows in Table 3). The training and test set sizes reduced to $20,721$ and $7,639$, respectively. We see that the discriminator performance for the text-level method with classification is now very close to when scores are excluded.

4.4 Effect on Natural Language Understanding

To understand the effect of each method on the NLU output, we computed the relative semantic error rate (SER), intent error rate, and slot error rate with respect to the reference data by calculating the

Table 4: Relative error rate change for in- and out-of-domain requests with respect to the reference.

	Relative Semantic Error Rate Change	**Relative Intent Error Rate Change**	**Relative Slot Error Rate Change**	**Relative Out of Domain Error Rate Change**
Real ASR Output	46.51%	58.24%	62.62%	2.65%
Audio-Level	71.61%	109.90%	76.78%	1.89%
Text-Level	65.72%	92.75%	62.43%	2.89%

exact intent and slot matches for in-domain and out-of-domain[5] requests (Table 4). This analysis was done on the 43% of the test data that was annotated for intent and slots. For the audio-level method, the higher relative SER change can be attributed to the higher number of word deletions compared to the the real ASR output. For the text-level method, 15% of the errors are related to one-word utterances. When an n-gram gets confused with a single word, it can create unrealistic entries in the confusion matrix for its subset words. Pruning the confusion matrix with phone level similarity, Levenshtein distance, or minimum frequency can reduce the unrealistic entries, but the remaining frequencies need to be re-weighted accordingly after pruning to maintain the same WER.

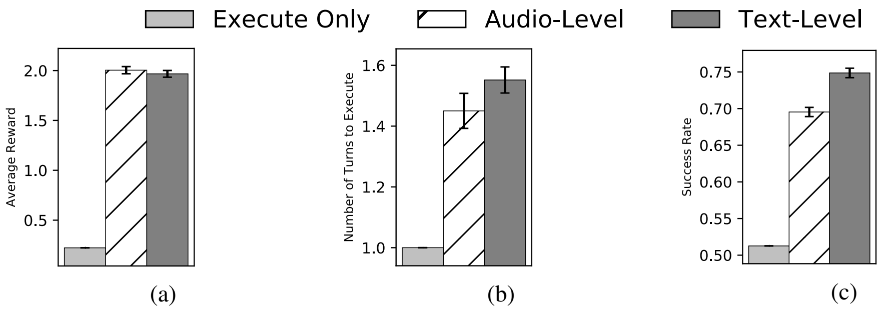
4.5 Comparison of Learned Dialog Policies

In this section we report preliminary results for learning dialog policies with a user simulator and each error simulation technique. We used a user simulator based on the approach presented in [8] and trained error recovery policies using deep RL. More specifically, the components used to learn the policies are as follows. The state is composed of 1) hypothesis intent and slot, 2) ASR confidence score, 3) previous agent action, 4) total number of clarifications so far in the dialog, and 5) number of clarifying questions for the current user request, where hypothesis is the agent's recognition. The action space is composed of three actions that are critical for conversational error recovery: *execute*, *confirm*, and *repeat*. *Execute* performs a backend action (e.g., accesses a database) and answers the user's question. Clarifying questions (*confirm*, *repeat*) are used to recover from errors. For the agent's policy, we used Dueling DDQN [34] with a fully-connected MLP to represent the deep Q-network. The hidden layers use a rectifier nonlinearity, and the output layer is a fully connected layer with linear activation function and a single output for each valid action. Finally, the reward function is summarized in Table 5. Here, *reference* indicates the actual intent and slot as generated by the user simulator, and *hypothesis* refers to what the agent understands. If *hypothesis* $\neq$ *reference*, this indicates that ASR/NLU errors were induced. The agents gets a positive reward for executing correctly, a negative reward for executing incorrectly, and smaller negative rewards for confirm and repeat. A successful execution is when the reference and hypothesis intent and slots are equal. The penalty for *repeat* is higher because it shows that the agent didn't understand any of what the user said. Additionally, if the user says an utterance with positive sentiment, the agent receives a small positive reward, but if the user interrupts the agent (a barge-in) or responds with an utterance with negative sentiment, the agent receives a small negative reward. Note that the state-based rewards are added to the action-based rewards. For more details see Appendix C.

[5] 23% of the annotated data are out-of-domain requests.

Table 5: Reward Function

a	s	r
execute(*hypothesis*)	$hypothesis = reference$	$+1$
execute(*hypothesis*)	$hypothesis \neq reference$	-1
confirm	$*$	-0.33
repeat	$*$	-0.50
$*$	positive sentiment	$+0.17$
$*$	negative sentiment	-0.17
$*$	barge-in	-0.17



(a)

(b)

(c)

Figure 3: Test performance of the dialog policies averaged over $2,400$ dialogs. a) average reward, b) number of turns to execute, and c) success rate.

To compare the policies learned with the audio- and text-level techniques, we used the user simulator with ASR output from real users and tested each agent on $2,400$ dialogs. To achieve a fair comparison, we fixed the random number generator so that the user simulator generated identical sequences of intents and utterances for both agents. As a baseline, we evaluated these dialogs with a policy that only executes. Figure 3 shows the results. The Y-axis in the three subfigures is the average reward, the average number of turns to execute, and the execution success rate of each technique, respectively. Average number of turns to execute is the number of clarification turns needed to execute for each agent, where 1 means execute without asking any clarifying question. Overall, policies learned using both audio- and text-level error simulators achieved significantly higher success rates compared to the execute-only policy, showing the importance of using a clarification capability. The text-level policy achieved a higher success rate, and was able to identify more incorrect turns (51%) compared to the audio-based policy (41%), however, this came at the cost of asking more clarifying questions.

We also conducted a user study to compare the real-world performances of the dialog policies learned with audio- and text-level techniques. To increase the statistical power given the limited number of subjects, we adopted the within-subjects experimental design and analysis [14], where each subject was assigned with and measured on all possible treatment combinations. More specifically, we recruited 16 internal subjects and considered two treatment factors: noise volume (high, low) and error simulation technique (audio-level, text-level), for a total of four unique treatment combinations. The sample size and number of treatment combinations are comparable to previously published studies under the same experimental design [22, 9]. We asked each subject to engage in four dialogs with MovieBot, one for each treatment combination. For example, if the treatment combination was "high noise volume + text-level technique", the subject would talk to the SLU application that adopted a dialog policy learned with the text-level error simulation technique, while a sound track of people talking on high volume was played in the background. We asked subjects to rate the quality of their most recent dialog on a 5-point scale at the end of each dialog. We followed the standard practice for conducting within-subjects experiments, including counter-balancing the order of treatment combinations. The cell and marginal means of the dialog ratings collected are shown in Table 6. Note that by design the observations from the same subject are correlated. We performed the two-way repeated measures ANOVA [14]. The partial output is shown in Appendix D. The audio- and text-level techniques have the same marginal mean ratings (3.09), hence there is 0 main effect for "technique" in the ANOVA. The audio-level technique has lower mean ratings than text-level for low noise volume, and vice versa for high noise volume. However, the differences are not statistically significant based on the p-value of the interaction effect of "noise × technique" in the ANOVA.

Table 6: Cell and marginal means of the dialog ratings

	Audio-level	**Text-level**	
Volume Low	3.06	3.38	3.22
Volume High	3.13	2.81	2.97
	3.09	3.09	3.09

Overall, the quantitative results from the user study do not reveal significant differences between the two techniques.

5 Conclusion

In this paper, we compared different error simulation techniques, with a focus on audio- and text-level methods, to assess their suitability for learning clarification dialog policies to recover from errors. In a variety of metrics considered, both the audio- and text-level methods produced similar results which are reasonably close to the ground truth. Therefore, the text-level method should be the preferred method for policy learning due to its simplicity and cheap computational cost.

Future work should focus on identification of types of errors that are recoverable, better detection of out-of-domain and out-of-vocabulary requests, and investigation of correlations between different sources of noise and the domain of user's requests. We plan to improve the existing audio-level method by inserting different levels and types of noise based on the audio content instead of uniform-randomly, such as inserting music noise when the user asks to change the song. Additionally, we plan to further improve the existing text-level method by leveraging phoneme/semantic similarity when mapping out-of-vocabulary words or trimming the n-gram confusion matrix. We can also explore generative adversarial networks [12] for simulating more realistic ASR output.

Acknowledgments

We would like to thank Shang-wen Li, Jin Cao and Jared Casale for helping set up the audio-based error simulation. We would also like to thank John Peach, Jason Pazis, Josep Valls Vargas, Lazaros Polymenakos, and Praveen Bodigutla for their feedback and support.

Appendix

A Schatzmann *et al.*'s (2007) N-Gram Confusion Matrix Approach

We briefly describe the text-level error simulation approach by Schatzmann et al. (2007), which served as the starting point for our approach described in Section 3.2:

- During training: For each reference utterance and ASR hypothesis pair from the training data, align the reference fragments (a "fragment" is essentially an n-gram for an arbitrary n) with hypothesis fragments by minimizing the Levenshtein distance. For each pair of aligned fragments, increment their frequency in the confusion matrix. Also, count the frequencies of all fragments in the training data.
- During testing: For each reference utterance in the test/new data, partition the sentence into fragments sequentially and probabilistically based on their frequencies in the training data: Moving from the first word to the last word in the utterance, a word belongs to the previous fragment with probability $freq$(previous fragment joined with the current word) / $freq$(previous fragment), otherwise it starts the next fragment. After partitioning, sample a replacement (including the original fragment in case of no error) for each fragment from the confusion matrix to generate the ASR hypothesis. Note that Schatzmann et al. did not mention how to handle out-of-vocabulary fragments. Lastly, sample an ASR score for the simulated ASR hypothesis from the training data ASR score distributions based on whether the simulated ASR hypothesis contains any word errors.

B Effect of Score on the Discriminator

In Section 4.3, we saw that (predicted) scores improve the accuracy of the discriminator. For text-level method with classification the reason is more subtle. Recall that in this approach we sample from the training set distribution within the classified score bin so there is no loss in granularity. As mentioned in Section 4.2, a certain hypothesis-reference pair U can have multiple duplicates with different real ASR scores, but the score model only predicts the most frequent bin. Now, let's say in the training set U appears 100 times in both real and simulated data, its real ASR scores are most frequently in bin 8, which happens 30% of the time. In this case, the score model always predicts bin 8 for

U. The discriminator learned the following rule - "if the score of U is in bin 8, U is simulated, and otherwise not simulated". The discriminator would then be able to correctly predict 100% of U in the simulated data and 70% of U in the real data, and achieve $(70 + 100)/200 = 85\%$ training set accuracy. Because the test set is similar to the training set, and we know that there are many hypothesis-reference pairs similar to U from the investigation of the spikes in Figure 2, it follows that the discriminator can have high test accuracy using the ASR scores as a predictor.

C Details of Policy Learning

We used Dueling DDQN [34] with a fully-connected MLP to represent the deep Q-network. The hidden layers use a rectifier nonlinearity, and the output layer is a fully connected layer with linear activation function and a single output for each valid action We trained the agent using an ϵ-greedy policy with ϵ decreasing linearly from 1 to 0.1 over $100,000$ steps. Additionally, we tuned a window size to include previous dialog turns as input and set γ to 0.97. We ran the method 30 times for $150,000$ steps, and in each run, after every 10,000 steps, we sampled 100 dialog episodes with no exploration to evaluate the performance. The optimal parameters were found using Hyperopt [2]. The learning parameters for each error simulator are show in the following table.

	Audio-Level	Text-Level
Number of Hidden Layers	2	2
Number of Hidden Nodes	64	128
Learning Rate	0.0001	0.0001
Dropout	0.1	0.5
Experience Replay Size	$10,000$	$15,000$
Embedding Size	1	20
Target Model Update Interval	$13,000$	$9,000$

D Two-Way Repeated Measures ANOVA on the Dialog Ratings in the User Study

Source	**Sum of Squares**	**Degrees of Freedom**	**F Value**	**p-value**
Noise Volume	1	1	1.071	0.317
Error Simulation Technique	0	1	0	1
Noise × Technique	1.562	1	3.641	0.076

References

[1] Layla El Asri, Jing He, and Kaheer Suleman. A sequence-to-sequence model for user simulation in spoken dialogue systems. *CoRR*, abs/1607.00070, 2016.

[2] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015.

[3] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. *CoRR*, abs/1810.05687, 2018.

[4] Heriberto Cuayahuitl, Steve Renals, Oliver Lemon, and Hiroshi Shimodaira. Human-computer dialogue simulation using hidden markov models. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 290–295, Nov 2005.

[5] Heriberto Cuayáhuitl. Simpleds: A simple deep reinforcement learning dialogue system. *CoRR*, abs/1601.04574, 2016.

[6] Wieland Eckert, Esther Levin, and Roberto Pieraccini. User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 80–87, Dec 1997.

[7] Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. Policy networks with two-stage training for dialogue systems. *CoRR*, abs/1606.03152, 2016.

[8] Maryam Fazel-Zarandi, Shang-Wen Li, Jin Cao, Jared Casale, Peter Henderson, David Whitney, and Alborz Geramifard. Learning robust dialog policies in noisy environments. *CoRR*, abs/1712.04034, 2017.

[9] Michael F. W. Festing. Randomized Block Experimental Designs Can Increase the Power and Reproducibility of Laboratory Animal Experiments. *ILAR Journal*, 55(3):472–476, 12 2014.

[10] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[11] Kallirroi Georgila, James Henderson, and Oliver Lemon. User simulation for spoken dialogue systems: learning and evaluation. In *INTERSPEECH*, 2006.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[13] Anjishnu Kumar, Arpit Gupta, Julian Chan, Sam Tucker, Björn Hoffmeister, and Markus Dreyer. Just ASK: building an architecture for extensible self-service spoken language understanding. *CoRR*, abs/1711.00549, 2017.

[14] David M. Lane, David Scott, Mikki Hebl, Rudy Guerra, Dan Osherson, and Heidi Zimmer. *Introduction to Statistics*. Online edition edition, 2013.

[15] Sungjin Lee and Maxine Eskenazi. Pomdp-based let's go system for spoken dialog challenge. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 61–66, Dec 2012.

[16] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing*, 8(1):11–23, Jan 2000.

[17] Xiujun Li, Zachary C. Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. A user simulator for task-completion dialogues. *CoRR*, abs/1612.05688, 2016.

[18] Xiujun Li, Yun-Nung Chen, Lihong Li, and Jianfeng Gao. End-to-end task-completion neural dialogue systems. *CoRR*, abs/1703.01008, 2017.

[19] Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Çelikyilmaz. Investigation of language understanding impact for reinforcement learning based dialogue systems. *CoRR*, abs/1703.07055, 2017.

[20] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2009.

[21] Olivier Pietquin and Thierry Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14:589–599, 2006.

[22] Ramani S. Pilla, David J. Kitska, and Catherine Loader. Statistical analysis of modified complete randomized designs: applications to chemo-orientation studies. *Journal of Experimental Biology*, 208(7):1267–1276, 2005.

[23] Silvia Quarteroni, Meritxell Gonzàlez, Giuseppe Riccardi, and Sebastian Varges. Combining user intention and error modeling for statistical dialog simulators. pages 3022–3025, 11 2010.

[24] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2real view invariant visual servoing by recurrent control. *CoRR*, abs/1712.07642, 2017.

[25] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowl. Eng. Rev.*, 21(2):97–126, June 2006.

[26] Jost Schatzmann, Blaise Thomson, and Steve Young. Error simulation for training statistical dialogue systems. In *2007 IEEE Workshop on Automatic Speech Recognition Understanding (ASRU)*, pages 526–531, Dec 2007.

[27] Konrad Scheffler and Steve Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 12–19, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[28] Edwin Simonnet, Sahar Ghannay, Nathalie Camelin, and Yannick Estève. Simulating asr errors for training slu systems. In *LREC*, 2018.

[29] Satinder P. Singh, Diane J. Litman, Michael J. Kearns, and Marilyn A. Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *CoRR*, abs/1106.0676, 2002.

[30] Matthew N. Stuttle, Jason D. Williams, and Steve J. Young. A framework for dialogue data collection with a simulated ASR channel. In *INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing, Jeju Island, Korea, October 4-8, 2004*, 2004.

[31] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *CoRR*, abs/1804.10332, 2018.

[32] Blaise Thomson, Milica Gasic, Matthew Henderson, Pirros Tsiakoulis, and Steve Young. N-best error simulation for training spoken dialogue systems. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 37–42, Dec 2012.

[33] Edmund Tong, Amir Zadeh, Cara Jones, and Louis-Philippe Morency. Combating human trafficking with multimodal deep models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1547–1556, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[34] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1995–2003. JMLR.org, 2016.

[35] Sihui Wang, Tom Gunter, David VanDyke, Thomas Voice, and Blaise Thomson. On modelling uncertainty in neural language generation for policy optimisation in voice-triggered dialog assistants. *2nd Workshop on Conversational AI: Today's Practice and Tomorrow's Potential, NeurIPS*, 2018.

[36] Longshaokan Wang. *Sufficient Markov Decision Processes*. PhD thesis, North Carolina State University, 2018.

[37] Jason D. Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393 – 422, 2007.

[38] Tiancheng Zhao and Maxine Eskénazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *CoRR*, abs/1606.02560, 2016.