# On Modelling Uncertainty in Neural Language Generation for Policy Optimisation in Voice-Triggered Dialog Assistants

**Sihui Wang***
Apple Inc.
Cambridge, UK
sihui_wang@apple.com

**Tom Gunter***
Apple Inc.
Cupertino, CA
tom_gunter@apple.com

**David VanDyke**
Apple Inc.
Cambridge, UK
dvandyke@apple.com

**Thomas Voice**
Apple Inc.
Cupertino, CA
tvoice@apple.com

**Blaise Thomson**
Apple, Inc.
Cambridge, UK
blaisethom@apple.com

## Abstract

While much effort has gone into user-modelling in the context of simulation for dialog policy training through reinforcement-learning (RL), the majority of this research has focused on matching user behaviour, with relatively little work dedicated to accurately replicating system uncertainty (system-error modelling) [14, 5]. In our opinion, this is unfortunate, as a primary benefit of the trained dialog policy is the ability to generate an appropriate user-prompt when the system is uncertain of the user's true goal, hopefully salvaging a chance at user-task completion [11, 21]. In this paper we describe and compare a set of existing and novel mechanisms for simulating voice-triggered dialog system uncertainty, in the context of turn-level neural language generation. We produce measures of divergence from the errors observed in the production system, suggest current best-practice, and propose directions for future investigation.

## 1 Introduction

Virtual assistants, such as Apple's Siri, Amazon's Alexa, or Google's Assistant, have seen growing commercial interest and have become a useful component of many people's lives. Researchers have suggested that using reinforcement learning could enable these systems to perform significantly better [7, 11]. In particular, reinforcement learning against a simulated user with simulated errors can enable these systems to respond more effectively in the context of noise, such as the speech recognition and natural language processing errors [21], which commonly occur in production systems.

Many early user simulators generated the user's utterances at a semantic level [14]. This means that instead of artificially generating "words" that a user could have said, user input was described with a higher level representation of meaning. Errors were typically also modelled at this semantic level.

When EncoderDecoder (Seq2Seq) models emerged as an effective method for generating language, many authors discovered that they can be applied to generating word-level responses in a dialog. Examples in the general dialog generation setting include [10, 8, 18, 16, 15], while [5] shows how these ideas can be extended to the case of goal directed dialog. There has, however, been limited research into adding error simulation to this setting, despite the fact that errors are intrinsic to any real world application. Indeed, much of the recent work assumes text input and makes no allowance

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

for the possibility of speech recognition errors. In addition, we recall that the state representation consumed by a dialog policy (i.e. the belief-state) is generally produced as a function of the complete beam of ASR uncertainty [21], so accurately modelling not only the most likely speech hypothesis, but in fact the whole beam is vital if we are to replicate the runtime environment.

In this paper we explore error simulation with these generative models of language in dialog, and believe that we are amongst the first to consider how to accurately emulate the full ASR beam.

In the context of goal driven semantic user simulators there is some related work to this approach of generating user input at the word level. The main example are simulators where the user's utterances are produced at a semantic level, and then the word level utterance is obtained by looking up a template [12, 2]. For some semantics, there will be no template available and then a generative model may be used [2]. Past word-level user simulators then approached the problem in one of three ways. The first is to generate audio via a Text-To-Speech (TTS) engine and then run it through an Automatic Speech Recognizer (ASR) [2]. The second is to simulate errors at the word-level via a translation model [13]. Finally, some authors have suggested simulating errors at the phone-level [19].

This paper compares five different approaches to error simulation in the context of a neural language generation system, drawing on the literature and extending where appropriate. In each case, we aim to emulate the output of the ASR, when presented with a generated sentence representing what the simulated user would have tried to enunciate. We experiment with:

1. Noise-Corrupted Synthesized Speech (NSS): TTS with additive Gaussian noise fed into the ASR component (where we have tuned the noise weighting to yield a word-error rate (WER) which matches the runtime system). This is similar to [3, 4, 2].
2. Word-Fragment Confusion (WFC) [13]: This simulates insertions, deletions and confusion of n-grams at the word level.
3. Learned Single Phone Confusion (SPC): A small extension of [19], also often seen in the speech research literature for fine-tuning discriminative language models [6]. Unlike the prior work we do not truncate the phone confusion matrix, and use the full language model to rescore.
4. Uniform Single Phone Confusion (UPC): A simplification of 3, where we do not fit the phoneme confusion—instead assuming it is uniform.
5. Phone Fragment Confusion (PFC): An extension of the phone-confusion model of [19], expanding the confusion matrix to cover insertion, deletion, and confusion of phone fragments (a combination of 2 + 3).

We apply each of the above error models to natural language utterances generated as part of a goal-based user simulation which is designed to emulate a live production dialogue system. We then evaluate these error models on their ability to match the distribution of errors found in the production usage logs.

In order to simulate user intents, the paper follows much of the literature in goal-based user simulation by separating the task into a goal generation step followed by a user dialog act generation step. Unlike previous approaches, the goal generation operates as a Probabilistic Context Free Grammar, with parameters trained from usage. The goal is broken up into user dialog acts by selecting some or all outstanding goal arguments to include, according to a hand-crafted model.

We go from user intent to language generation by taking the approach of [20] applied within the context of user simulation. A semantically conditioned decoder network generates tokens one by one until a full sentence is completed. Compared to other word-level user simulators for goal-based dialog [5], this has an advantage of allowing the separate generation of goals, intents and words. This separation helps with model interpretability, and makes it easier to integrate domain knowledge and simplify the transfer to unseen domains.

By connecting our user simulation to the live production system, we can generate realistic dialogues which emulate real user interactions. We incorporate each candidate error model in turn and then evaluate how closely the corresponding output matches the distribution of errors observed in the usage logs. Our evaluation is based on BLEU [9] scores and on the output of neural discriminators which we fit to distinguish between the real and fake ASR beams. This paper represents an early step fowards in finding realistic error models for word-level based user simulators. This is a topic that we feel is of vital importance in order to trust that dialogue managers trained on user simulations can perform well when interacting with real users.

## 2 User-simulation

We describe a simulator which constructs a hierarchical user-goal, before breaking it into turn level semantics. At each turn we then use the NLG system to convert the semantics into plain text, before applying error simulation. The overall 'user-model' we use is broken up as follows.

### 2.1 A goal-model

We need to generate goals in keeping with what is seen in the production usage log. For the purposes of this work, we use heuristics to automatically extract a set of final-goals from the usage log, and then fit a probabilistic context-free grammar (PCFG) [1] at the node level using this goal set (we also experimented with sampling from the goal set directly). To generate a sample goal, we start with a root node and sample according to the PCFG.

### 2.2 A user-turn model

This takes a goal and breaks it down into per-turn segments. E.g. a user could fully specify all arguments of a goal in one turn, or they might break the goal into sub-arguments. Due to a dearth of multi-turn interactions in current usage distribution, we simply hand-craft this model at the whole-goal versus argument-per-turn level, based on statistics estimated from the usage log (i.e. does the simulator speak the whole goal at once or not).

### 2.3 A Natural Language Generation (NLG) model

This generates a natural language description of the semantic intent produced by the user-turn model. Generated language should be similar to that found in live production usage (in a distributional sense). We base our NLG model on [20], wherein the key innovation is to augment a Long-Short Term Memory (LSTM) cell with an additional memory unit, which keeps track of which semantic concepts from the input intent have been 'mentioned'—this is termed a SC-LSTM.

This model consumes as input a fixed dimensional vector, which is the result of featurising the turn-level user-act. We then generate *delexicalised* text conditional on this user-act feature [20]. As a result, the difficulty of our modelling problem is significantly reduced, with a vocabulary size of just $2,050$ tokens including special tokens and punctuation.

To fit this model we use three months of production data, keeping only examples where the speech transcription was high confidence (so as to be reasonably secure in the understood semantic content). We split the data into training, validation and testing sets in the ratio 70%/20%/10%. On a uniformly sampled subset of $100,000$ examples from the test set, we observe $5,014$ unique de-lexicalised semantic inputs to the NLG component. This same test set yields an average sentence-level BLEU score of $94.44$, and a corpus-level BLEU of $80.89$ (comparing against the top-20 reference's found in usage data). In addition, we calculate the fraction of examples for which we correctly convey all semantic meaning without generating any extra semantic fields. This occurs $78.76\%$ of the time. These figures suggest that the NLG model is performing well in absolute terms, although the reader should mentally calibrate against the small vocabulary size and large reference set for each semantic input. A template based system would, of course, perform very well on semantic score, but would not easily scale across goals, nor would it produce the variety of language that real users produce. Also the lack of variability in template generation would require a further layer of semantic error simulation in addition to the acoustic error simulation we consider in Section 3. We also note that these metrics are not conclusive for NLG, as described in [17].

## 3 Error Models

Our goal is to faithfully model system errors in the context of a turn level utterance generated by the NLG system. It is our assumption, following [13], that if we are able to effectively simulate at the level of a component, then downstream error models are not required, and instead are fulfilled by

| NLG Output Beam: Semantics A | NLG Output Beam: Semantics B | NLG Output Beam: Semantics C |
|---|---|---|
| 1. “Do I have school tomorrow?” | 1. “What appointments do I have this week?” | 1. “What happens at 4?” |
| 2. “Do we have school tomorrow?” | 2. “What’s on my calendar this week?” | 2. “Siri what happens at 4?” |
| 3. “Siri do we have school tomorrow?” | 3. “What’s on my calendar for this week?” | 3. “What happens at 4 o’clock?” |
| 4. “Siri do I have school tomorrow?” | 4. “What are my appointments this week?” | 4. “What’s on at 4?” |
| 5. “Will we have school tomorrow?” | 5. “Show me my calendar for this week” | 5. “What is on at 4?” |
| 6. “Will I have school tomorrow?” | 6. “What’s on my schedule this week?” | 6. “How about 4 o’clock?” |

Figure 1: We show examples of the NLG output beam for three different semantic inputs. Note that these are before phonetic noise is introduced, so we expect them to be fluent and diverse sequences, all of which describe the same semantics (for each of A, B and C).

executing the runtime system on the simulated input. This means, for example, that we do not need to in-addition model semantic errors.

We draw on and extend several alternatives presented in the prior literature.

### 3.1 Noise-Corrupted Synthesized Speech (NSS)

We start with perhaps the simplest alternative, previously studied in various guises by [3, 4, 2]. It uses the runtime TTS system to generate each user utterance in spoken form and then adds Gaussian noise scaled so as to produce an average WER matching that of the production system. The corrupted audio stream is then input to the ASR component and then on to the rest of the system.

The downside of this approach is that it is slow, although this is not neccessarily a show-stopping problem—for downstream usage we can scale over many dialogs in parallel.

Its main benefits lie in simplicity of implementation, (assuming access to performant TTS and ASR systems), and ease of transfer to new languages and domains (again, given access to the appropriate runtime components). We leave it to a later work to investigate more sophisticated additive noise systems for TTS, as there is limited prior art.

### 3.2 Word-Fragment Confusion (WFC)

We implement [13]. In this paper, ASR confusions are viewed as translations of a source utterence $w$ to a confused target utterance $\tilde{w}$. Each such utterance is a sequence of $S$ words, $(w_i)_{i=1}^{S}$, or equivalently a sequence of $L$ fragments $(f_k)_{k=1}^{L}$, where each fragment is a contiguous N-gram from $w$. We break the target utterance up in a similar fashion, without loss of generality make the assumption that both source and target have the same number of confusable fragments.

We follow the notation from [13], and define a mapping function $\gamma$ which maps word indices to fragment indices. As an example, $\gamma = (\gamma_k)_{k=1}^{S}$ with $\gamma_k = i$ if the $k$'th word in the source utterance $w$ belongs to the source fragment $f_i$. Similarly for the target $\tilde{\gamma}$. We can write the modelling objective as finding the conditional distribution:

$$P(\tilde{w}, \tilde{\gamma}, \gamma | w) = P(\tilde{w}, \tilde{\gamma}, | \gamma, w) P(\gamma | w). \tag{1}$$

An assumption is made that the alignment of each word depends only on words and fragments seen prior in the sentence, yielding the following:

$$P(\gamma | w) = P(\gamma_1 | w_1) \prod_{i=2}^{S} P(\gamma_i | w_i, (w_j)_{j=start}^{i-1}, \gamma_{i-1}), \tag{2}$$

where $(w_j)_{j=\text{start}}^{i-1}$ are the words assigned to $\gamma_{i-1}$. Word-fragment assignment probabilities evolve according to:

$$P(\gamma_i|w_i, (w_j)_{j=start}^{i-1}, \gamma_{i-1}) = \begin{cases} \phi; & \text{if}, \gamma_i = \gamma_{i-1}, \\ 1-\phi; & \text{if}, \gamma_i = \gamma_{i-1}+1, \\ 0; & \text{otherwise.} \end{cases} \tag{3}$$

$\phi$ represents an empirical likelihood of seeing $w_i$ follow $w_{i-1}$ in the current fragment.

Given an utterance and a empirical frequency estimate for each N-gram, we may sample a partition (i.e. a fragment set) in a generative fashion, with only a single pass through the given word sequence.

#### 3.2.1 Aligning source and target utterances

If $\gamma$ is known, the conditional probability of $\tilde{w}$ may be derived using just the fragment confusion probabilities:

$$P(\tilde{w}, \tilde{\gamma}|\gamma, w) = P((\tilde{f}_i)_{i=1}^{L}|(f_i)_{i=1}^{L}) \tag{4}$$

$$\approx P((\tilde{f}_i)_{i=1}^{L}) \prod_{i=1}^{L} P(\tilde{f}_i|f_i). \tag{5}$$

When simulating errors, we can exploit the conditional independence approximation in Equation 5 by over-sampling and then making use of the fact that $P((\tilde{f}_i)_{i=1}^{L}) = P(\tilde{w})$ to allow us to score and rank the alternatives using the runtime language model.

The confusion statistics are collected by aligning source and target training data, using a dynamic programming algorithm equipped with a Levenshtein distance metric and fixed cost for insertion, deletion and substitution.

All of the above are estimated using one day of production data.

### 3.3 Learned Single Phone Confusion (SPC)

We compile the token-sequence sampled from the NLG model into a word-level Finite State Transducer (FST), before further mapping to a phone level FST by composing with a speech-lexicon dictionary. We then compose this with a learned phone confusion FST, and take the $50$ shortest-weighted paths to produce output sequences. We then invert this into a word-token representation, and make use of the language-model to rerank the generated alternatives.

This closely resembles the approach taken in [19], except we do not truncate the phone confusion matrix, and apply the full language model to rerank the noisy examples.

The phonetic confusion is fit by considering one day of production data: for each pair of the $K$-best phone sequences, we align the sequences using a dynamic programming algorithm with substitution, insertion, and deletion costs of $1$. We then estimate an empirical probability of phone-to-phone confusion. While this ignores phone-sequence effects, we did not find a great deal of additional benefit to considering higher order phone confusions (see Section 3.5).

### 3.4 Uniform Single Phone Confusion (UPC)

We proceed as in Section 3.3, except that we simply impose uniform phone substitution probabilities.

As before, we overgenerate and then rescore and rerank with the language model. We include this benchmark as a lightweight ablation test to demonstrate the relative power of overgeneration and rescoring with the language model.

### 3.5 Phone-Fragment Confusion (PFC)

We combine the concepts discussed in Section 3.2, and Section 3.3, producing a ‘phone-fragment’ confusion model. The idea behind this is to attempt to capture larger confusable structures (similar to

Table 1: Average BLEU score as computed against reference ASR beam.

| Top N | NSS | WFC | SPC | UPC | PFG |
|---|---|---|---|---|---|
| N = 1 | 82.3 | 76.9 | 93.7 | 96.5 | 83.3 |
| N = 2 | 60.5 | 77.6 | 77.6 | 74.0 | 67.1 |
| N = 3 | 46.9 | 77.1 | 71.4 | 70.5 | 63.0 |
| N = 4 | 39.8 | 74.7 | 70.6 | 69.5 | 58.3 |
| N = 5 | 35.9 | 73.8 | 68.7 | 68.4 | 57.3 |

the word-fragment confusion model), but by doing so at the phonetic level avoid poor performance if NLG produces an infrequently seen yet still fluent sentence.

The model and training proceeds very much as in Section 3.2, except we consider phones instead of words and phone-fragments rather than word-fragments. Once again, all aspects of the model which need to be fit consider one day of production data.

## 3.6 Comparison

As all of our listed mechanisms for noise-modelling either produce input to ASR, or model the output of ASR, it makes most sense to evaluate performance at the output of the ASR component.

We are interested in three aspects of performance:

1. At an utterance level does the system generate language which is well matched to the 'reference set' of the observed ASR confusion?
2. Is the system able to overgenerate, such that we can separately model a truncation process so as to yield the same distribution of N-best counts as the live system?
3. As we increase the beam of noised alternatives, how closely do we match (in distribution) the 'reference set'?

To explore the above, we run two simple experiments. In each, we seed the noise generation mechanism with the top alternative from a production log (unseen at training time), overgenerate (where possible), and rerank with the language model—retaining the top-5 alternative speech hypotheses. For this analysis, we sidestep the issue of truncating the number of generated hypotheses to match the distribution of speech alternative counts, as this is shared amongst all methods barring synthesized speech. We presume that so long as we are able to over-generate we may produce a variety of models to truncate the beam so that the hypothesis-count distribution matches that seen in the logs.

### 3.6.1 Fluency of generated alternatives

In order to give some (objective as well as relative) measurement of performance, we rank each set of alternatives using the language model, and then take an average of 1-to-4-gram BLEU [9] scores, as compared to the reference set of ASR alternatives. We do this for each of the (up to top-5) alternatives, and take an average of the BLEU scores as computed for each alternative against the reference set.

We note that $\sim 25\%$ of the 'noise-corrupted synthesised speech' test examples only yielded a single output at the level of the ASR beam, meaning that they are overly penalized as we drop down the table (we assume a BLEU of 0.0 if the method is unable to produce an alternative at that depth). Even at the level where the method is not punished however, it performs poorly under this metric, suggesting that if it is to be used for policy training a more sophisticated noise model should be fit, and perhaps to a more complicated measure than WER.

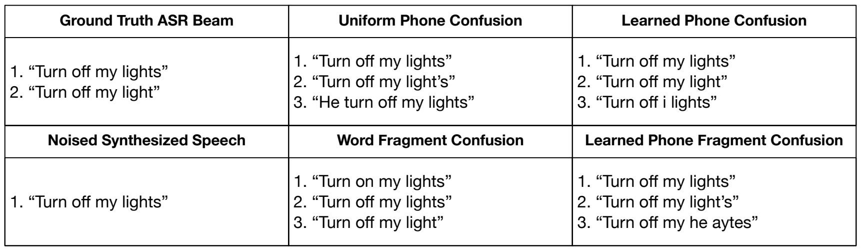
The power of the language model (under this metric at least) is illustrated by the high performance of a simple uniform confusion model. In practice, we see worse performance (c.f. the learned single phone confusion model), as it generates a wider range of alternative hypotheses, some of which may score highly under the language model.

Finally, we can see that the word-fragment model performs surprisingly stably, and in particular exceeds the other methods according to BLEU score as we drop down the list. A qualitative assessment suggests that looking at BLEU alone may obscure some drawbacks of the method (see

Figure 2), whereby it produces *fewer* out of reference tokens, but they are less phonetically reasonable than the reference set (or the single phone confusion model).

The word-fragment model also performs well on long named-entities, which are low-medium usage, often confused, and too long to be accurately modelled by the single-phone confusions.

Some examples of generated confusion are shown in Figure 2.



| Ground Truth ASR Beam | Uniform Phone Confusion | Learned Phone Confusion |
|---|---|---|
| 1. “Turn off my lights”<br>2. “Turn off my light” | 1. “Turn off my lights”<br>2. “Turn off my light’s”<br>3. “He turn off my lights” | 1. “Turn off my lights”<br>2. “Turn off my light”<br>3. “Turn off i lights” |
| **Noised Synthesized Speech** | **Word Fragment Confusion** | **Learned Phone Fragment Confusion** |
| 1. “Turn off my lights” | 1. “Turn on my lights”<br>2. “Turn off my lights”<br>3. “Turn off my light” | 1. “Turn off my lights”<br>2. “Turn off my light’s”<br>3. “Turn off my he aytes” |

Figure 2: We show example beams from each of the proposed noise generation mechanisms, as well as the ‘ground truth’ sampled from logs.

### 3.6.2 Distributional overlap with the reference set

BLEU score provides one, very coarse, measure of distributional overlap between the generated alternatives and the reference set. Defining a formal measure of divergence directly on the N-best lists is non-trivial, however, so we opt to fit a discriminator between each alternative noise model and the ground truth hypothesis set.

We posit that the ease with which the descriminator is able to seperate real from fake ASR beams is indicative of how strongly overlapping the two distributions are. In theory, a good measure of distributional overlap between the generated ASR output and ground truth gives us an indication as to whether fitting a dialog policy using this error model will be monotonically better or not. It does not, however, tell us how much of an improvement there will be—for that we need post-hoc (and probably human) evaluation of the trained dialog policies–something we do not cover in this work.

The discriminator architecture is kept reasonably simple for this exploratory analysis: we pad to the top $5$ alternatives, and each alternative is extended to contain $21$ tokens. $2$ dimensional word embeddings are inferred, and these are then sum-pooled per-hypothesis, before being concatenated together and fed into a dense binary classifier (i.e. we are order invariant for tokens, but order sensitive to hypotheses).

We also experiment with using LSTM encoders with 2 dimensional hidden cells (in place of the ‘bag-of-words’ sum-pool approach). This gives us some insight into how different the sequences of words (as opposed to simple sets) are for each type of artificially generated beam. Results for this experiment are in Figure 4.

For all discriminators we use batch sizes of $256$, and a learning rate of $4e^{-4}$, evaluating on $10\%$ of the data after every epoch. Models are deliberately kept (very) low capacity for these initial experiments.

Clearly significantly more work should be invested in discriminator architectures and training mechanisms. An early takeaway, however, appears to be that PFC is by far the most easily discriminated, while WFC is hardest to separate for both the bag-of-words and LSTM based discriminators. In general, we find that NSS is more easily discriminated at the very start of training (there is a strong signal in the difference of hypothesis counts per instance), however harder to fully separate at the end of training (implying that there is more distributional overlap with the true ASR beams, at least given the constraints of this architecture). The fact that SPC drops down the rankings when we take word order into account, however, suggests that this approach produces plausible words, but sometimes in unusual relative positions within the utterance. This makes intuitive sense, and is something that might be addressed by a more structured phonetic confusion, however our experiments with PFC have proved dissapointing.

We note that this experiment suggests that although some error models are marginally easier to discriminate from the reference beam set than the alternatives, by far the largest takeaway is how

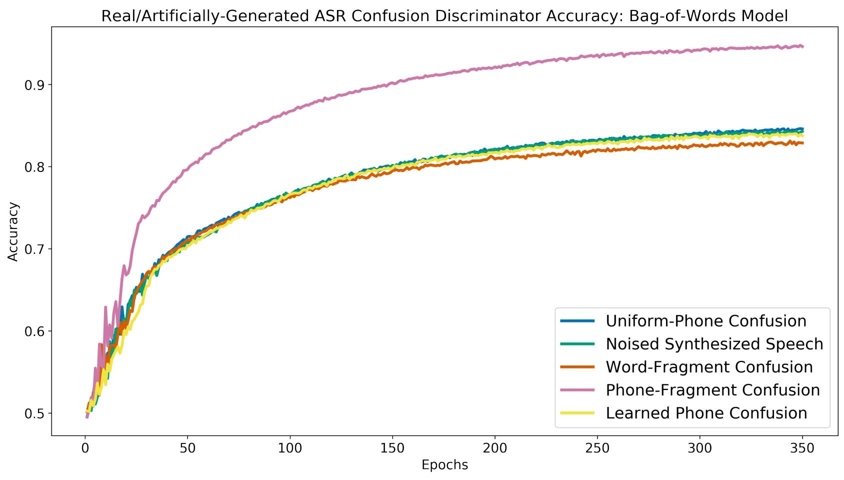


Figure 3: Accuracy of a bag-of-words discriminator model at training time, fit to determine whether the list of ASR hypotheses presented are artifically generated or real (i.e. harder to discriminate implies a more realistic ASR error simulation model).

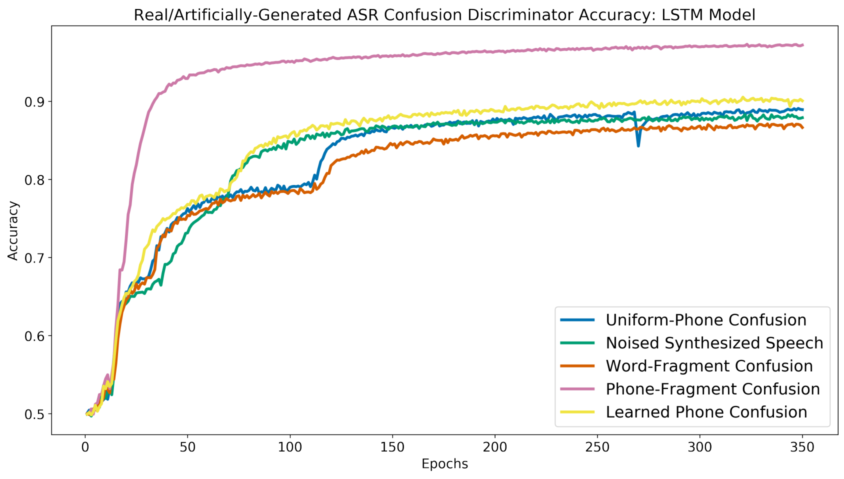


Figure 4: Accuracy of a LSTM discriminator model at training time, fit to determine whether the list of ASR hypotheses presented are artifically generated or real.

easily all the noise models may be discriminated from ‘real’ ASR beams. This indicates that we are still a way away from being able to accurately model system uncertainty, implying that this is an area where further research effort should be applied.

## 4 Conclusion

We have noted that relatively little of the literature on user-modelling for policy optimisation in statistical dialog systems focuses on accurately modelling system errors (including ASR input). We have embarked on an initial exploration of the prior art, and made a few small extensions. We note the power of over-generation and rescoring with the full language model, and more importantly highlight the failure of any of the existing techniques (including our extensions proposed herein) to produce ASR hypotheses which approach strong distributional overlap with those collected by a live system interacting with real users. We propose that accurately modelling system uncertainty is important if we are to trust usersimulator-trained dialog policies to interact with real users on complex domains, and intend to extend on this line of work in the near future, noting that it is important to give results as measured according to the performance of the trained dialog policy.

**Acknowledgments**

We thank Steve Young, Thor Helgason, and Vera Sheinman for useful feedback on drafts of this work. We are also grateful to Joris Dreisen for several helpful discussions about acoustic modelling.

## References

[1] Zhiyi Chi and Stuart Geman. Estimation of probabilistic context-free grammars. *Comput. Linguist.*, 24(2):299–305, June 1998.

[2] 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.

[3] Mark J. F. Gales, Anton Ragni, H. AlDamarki, and C. Gautier. Support vector machines for noise robust asr. *2009 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 205–210, 2009.

[4] Alexander Gutkin, Linne Ha, Martin Jansche, Knot Pipatsrisawat, and Richard Sproat. Tts for low resource languages: A bangla synthesizer. In *10th edition of the Language Resources and Evaluation Conference, 23-28 May 2016*, pages 2005–2010, Portorož, Slovenia, 2016.

[5] Florian Kreyssig, Inigo Casanueva, Pawel Budzianowski, and Milica Gasic. Neural user simulation for corpus-based policy optimisation for spoken dialogue systems. In *Proceedings of the 19th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, 2018.

[6] Gakuto Kurata, Abhinav Sethy, Bhuvana Ramabhadran, Ariya Rastrow, Nobuyasu Itoh, and Masafumi Nishimura. Acoustically discriminative language model training with pseudo-hypothesis. *Speech Commun.*, 54(2):219–228, February 2012.

[7] E. Levin, R. Pieraccini, and W. Eckert. Using markov decision process for learning dialogue strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1998.

[8] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 285–294, Prague, Czech Republic, September 2015. Association for Computational Linguistics.

[9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[10] Alan Ritter, Colin Cherry, and William B. Dolan. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 583–593, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[11] Nicholas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 93–100, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[12] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, NAACL-Short '07, pages 149–152, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[13] Jost Schatzmann, Blaise Thomson, and Steve J. Young. Error simulation for training statistical dialogue systems. In *ASRU*, pages 526–531. IEEE, 2007.

[14] 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.

[15] Iulian Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues, 2017.

[16] Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models.

In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 3776–3783. AAAI Press, 2016.

[17] Shikhar Sharma, Layla El Asri, Hannes Schulz, and Jeremie Zumer. Relevance of unsupervised metrics in task-oriented dialogue for evaluating natural language generation. *CoRR*, abs/1706.09799, 2017.

[18] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205. Association for Computational Linguistics, 2015.

[19] 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.

[20] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-hao Su, David Vandyke, and Steve J. Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *CoRR*, abs/1508.01745, 2015.

[21] Steve Young, Milica Gasic, Blaise Thomson, and Jason Williams. Pomdp-based statistical spoken dialogue systems: a review. pages 1–20, January 2012.