A Good Sample is Hard to Find: Noise Injection Sampling and Self-Training for Neural Language Generation Models

Chris Kedzie
Columbia University
Department of Computer Science
kedzie@cs.columbia.edu

Kathleen McKeown
Columbia University
Department of Computer Science
kathy@cs.columbia.edu

Abstract

Deep neural networks (DNN) are quickly becoming the de facto standard modeling method for many natural language generation (NLG) tasks. In order for such models to truly be useful, they must be capable of correctly generating utterances for novel meaning representations (MRs) at test time. In practice, even sophisticated DNNs with various forms of semantic control frequently fail to generate utterances faithful to the input MR. In this paper, we propose an architecture agnostic self-training method to sample novel MR/text utterance pairs to augment the original training data. Remarkably, after training on the augmented data, even simple encoder-decoder models with greedy decoding are capable of generating semantically correct utterances that are as good as state-of-the-art outputs in both automatic and human evaluations of quality.

1 Introduction

Deep neural network (DNN) architectures have become the standard modeling method for a host of language generation tasks. When data is plentiful, the sequence-to-sequence framework proves to be incredibly adaptable to a variety of problem domains. Recent evaluations of end-to-end trained DNNs for dialogue generation have shown that they are capable of learning very natural text realizations of formal meaning representations (MRs), i.e. dialogue acts (DAs) with slot-filler type attributes (see Figure 1 for an example). In many cases, they beat rule and template based systems on human and automatic measures of quality (Dušek et al., 2019).

However, this powerful generation capability comes with a cost; DNN language models are notoriously difficult to control, often producing quite fluent but semantically misleading outputs. In order for such models to truly be useful, they must be capable of correctly generating utterances for novel MRs at test time. In practice, even with delexicalization (Dušek and Jurčíček, 2016; Juraska et al., 2018), copy and coverage mechanisms (Elder et al., 2018), and overgeneration plus reranking (Dušek and Jurčíček, 2016; Juraska et al., 2018), DNN generators still produce errors (Dušek et al., 2019).

Inform	name[The Golden Curry]
	near[The Six Bells]
	familyFriendly[yes]

Training Reference Utterance
Near The Six Bells is a venue that is children friendly named The Golden Curry.

Figure 1: Example MR for the Inform DA with example human reference utterance.

In this work, rather than develop more sophisticated DNN architectures or ensembles, we explore the use of simpler DNNs with self-training. We train a bare-bones unidirectional neural encoder-decoder with attention (Bahdanau et al., 2014) as our base model from which we sample novel utterances for MRs not seen in the original training data. We obtain a diverse collection of samples using noise injection sampling (Cho, 2016). Using an MR parser, we add novel utterances with valid MRs to the original training data. Retraining the model on the augmented data yields a language generator that is more reliable than the sophisticated DNNs that have been recently developed, in some cases reducing test set semantic errors to zero, without sacrificing linguistic quality.

In this paper we make the following contributions. 1) We propose a general method of data augmentation for natural language generation (NLG) problems using noise injection sampling and self-training. 2) We show a reduction of attribute realization errors across several dialog generation datasets, while achieving competitive automatic and human quality evaluation scores. 3) Finally, we show that these results hold even when the MR parser is noisy or we use fully lexicalized generation models.[1]

2 Datasets and Problem Defintion

We ground our experiments in three recent dialogue generation datasets, the E2E Challenge Dataset (Dušek et al., 2019), and the Laptops and TVs datasets (Wen et al., 2016). We only briefly review them here. Each dataset consists of dialog act MRs paired with one or more reference utterances (see Figure 1 for an example from the E2E dataset). The structure of each MR is relatively simple, consisting of the dialog act itself, (e.g. *inform*, *recommend*, *compare*, etc.) and a variable number of attribute slots which need to be realised in the utterance. All attribute values come from a closed vocabulary. If an attribute is not present in the MR it should not be realized in the corresponding utterance.

The three datasets also represent different training size conditions; there are 42,061, 7,944, and 4,221 training examples in E2E, Laptops, and TVs datasets respectively.

The NLG task for all three datasets is to produce an utterance for a given MR such that all attributes in the MR are realized naturally and correctly.

E2E Model Input Following previous sequence-to-sequence approaches for the E2E dataset (Juraska et al., 2018), we treat the MRs as a linear sequence of tokens $x = (x_1, \ldots, x_8)$ where each of the 8 positions represents the value of a corresponding attribute. If an attribute is not specified in the MR we assign it an attribute specific *n/a* token. In the E2E dataset there is only one dialog act type, *Inform*, so we do not represent it in x.

Prior work often delexicalizes the *Name* and *Near* attributes (i.e. replaces verbalizations of attribute values with a placeholder token), which can later be replaced with the original attribute value in a post-processing step. For example, the delexicalized version of the utterance in Figure 1 would be "Near NEAR is a venue that is children friendly named NAME." *Name* and *Near* have a relatively large vocabulary of valid slot fillers, some of which are only seen infrequently in the training data; it can be difficult for fully lexicalized models to produce some of the rarer location names for these attributes.

However, since delexicalization might be difficult or impossible in other domains, we implement both delexicalized and lexicalized versions of the generation models on the E2E dataset to more fully evaluate the self-training method.[2]

Laptops and TVs Model Inputs The Laptops and TVs datasets have a more diverse set of dialog acts and can have repeated attributes (with different values) in some cases, so we abandon our fixed length, fixed position encoding, and represent each MR as a initial dialog act token and then a variable length sequence of tokens for each of the specified attributes. The evaluation script for these datasets uses delexicalization to evaluate attribute realization error, and so we use it here to be consistent with prior work, delexicalizing all possible attributes. See Appendix B for example input sequences for all datasets.

3 Generation Model

We treat the generation task as a sequence-to-sequence transduction problem, where we learn a probabilistic mapping p from the linearized MR x to a sequence of N tokens $y = (y_1, \ldots, y_N)$ consituting the utterance. The model p is implemented using a two-layer unidirectional[3] encoder-decoder architecture with gated recurrent units (Cho et al., 2014) and feed-forward style attention (Bahdanau et al., 2014) as this is a canonical recurrent architecture for sequence-to-sequence modeling. We use 512 dimensions for all embeddings and GRU states.

We fit the model parameters θ by minimizing the negative log likelihood of the training set $\mathcal{D}$, i.e. $\mathcal{L}(\theta) = -\sum_{(x,y)\in\mathcal{D}} \log p(y|x;\theta)$ using stochastic gradient descent. Going forward we omit θ for clarity.

3.1 Generating from p

Deterministic Decoding Given an arbitrary MR x, we can generate an utterance $\tilde{y}$ using greedy decoding, i.e. $\tilde{y}_i = \arg\max_{y_i} p(y_i|\tilde{y}_{<i}, x)$.

[1]Code and data for this paper can be found at https://github.com/kedz/noiseylg

[2] Additional preprocessing details can be found in Appendix A.

[3]In initial experiments, we found the unidirectional encoder to perform better than a bidirectional one.

To produce an "n-best" list of outputs, we can also use beam decoding where n candidate utterances are maintained during each decoding step. Both greedy and beam decoding are known to produce somewhat homogeneous outputs (Serban et al., 2016). Diversifying beam outputs often involves careful tuning of secondary search objectives which trade off fluency (Li et al., 2015).

Moreover, when training fully lexicalized models we found that we could often *not* produce certain *Name* and *Near* attribute values. For example, we constructed a novel MR with *near[Burger King]* and fed it into our base generator. Even with an impractically large beam size of 512, we could not produce an utterance with "Burger King" in it. This failure mode makes beam search a relatively unuseable method for producing utterances for MRs under-represented in the training data.[4]

To overcome this limitation we explored several sampling methods for generating these rarer utterances, namely ancestral and noise injection sampling.

Stochastic Decoding Ancestral sampling (i.e. drawing a random token from $p(Y_i|y_{<i}, x)$ at each decoder step) is another option for generating diverse outputs, but the outputs can be of lower fluency and coherence. Producing rare tokens often involves tuning a temperature parameter to flatten the output distribution, but this again can hurt fluency.

As an alternative, we can obtain diverse samples from greedy decoding by injecting Gaussian noise into the decoder hidden states following Cho (2016). Under our model, the probability $p(y_i|y_{<i}, x) = f(x, h_i)$ of the i-th token is a function of the encoder inputs x and the decoder hidden state $h_i \in \mathbb{R}^D$. When peforming noise injection sampling, we replace h_i with a noisy state $\tilde{h}_i = h_i + \epsilon_i$ where ϵ_i is drawn from a D-dimensional Gaussian distribution $\mathcal{N}(0, \sigma_i^2 I)$ with variance $\sigma_i^2 = \frac{\sigma_0^2}{i}$. The base variance σ_0^2 is a a hyperparameter. Effectively, the first few steps allow the decoder to reach a novel hidden state space, while the gradually diminishing noise allows the decoder to produce fluent outputs.

Generating Rare Values Remarkably, the samples obtained by noise injection maintain fluency and valid syntactic structure. At the same time they often hallucinate or drop attributes. For example, see the last noise injection sample in Table 1 where the attribute *near[The Bakers]* is hallucinated. This kind of error is actually useful because, as long as we have a reliable MR parser, we can recover the MR of the sample and we now have a totally valid extra datapoint that we could use for training. Syntactic errors of the kind produced by ancestral sampling (see the last ancetral sampling example in Table 1), on the other hand, are not good to train on because they damage the fluency of the decoder.

Returning to the "Burger King" example, with noise injection sampling we were able to produce over 10,000 novel instances of it. See Appendix D for more samples.

4 MR Parsing Model

Given a novel utterance $\tilde{y}$ sampled from p, we need to reliably parse the implied MR, i.e. $\tilde{x} = q(\tilde{y})$, where q is our parsing model. We have two things going for us in our experimental setting. First, even with noise injection sampling, model outputs are fairly patterned, reducing the variability of the utterances we need to parse in practice.

Second, the MRs in this study are flat lists of attributes that are somewhat independent of each other. We only need to detect the presence of each attribute and its value. For the Laptops and TVs datasets we also need to recover the dialog act but these also are signaled by a fairly limited repertoire of cues, e.g. "we recommend." Given this, we experiment with both hand crafted regular expression rules and learned classifiers to predict the value of an attribute if present or that it is missing.

Rule-based parser $q_{\mathfrak{R}}$ We design hand-crafted regular expression based rules to match for the presence of key phrases for each of the attributes and DAs in the datasets while also checking to make sure that there is only one match per attribute.

To construct the rules, we look through both the training data references as well as the generation model outputs as this is what the rules will be operating on in practice. For each lexicalized attribute (and DA) we develop a list of regular expressions such as, `/is (family|kid|child) friendly/` $\Rightarrow$ *familyFriendly[yes]*. For the delexicalized attributes, we simply check for the presence of the placeholder token.

[4]The MR in this case had three attributes. *near[Burger King]* only occurs in size eight MRs in the training data.

Input MR
Inform(name[The Cambridge Blue], eatType[Restaurant], customerRating[high], food[Italian])
Ancestral Sampling
The Cambridge Blue is an Italian restaurant with a high customer rating.
The Cambridge Blue is an Italian restaurant with high ratings.
*Italian restaurant, the Cambridge Blue, has a high customer rating. (Phrase fragments, not fluent.)
Noise Injection Sampling
The Cambridge Blue is a restaurant that serves Italian food. it has a high customer rating.
*The Cambridge Blue is a highly rated restaurant. (Drops *food[Italian].*)
*The Cambridge Blue is a restaurant located near the Bakers. (Hallucinates *near[The Bakers].*)

Table 1: Examples of ancestral sampling and noise injection sampling ($\sigma_0 = 1.0$). * indicates output that is either not grammatical or is not correct with respect to the input MR. Text in parentheses explains the details of the error in either case.

We design these rules to be high precision, as it is safer to miss out on more obscure varieties of utterance to avoid adding incorrectly parsed data points. However, in many cases the rules are also high recall as well. The average F-score on the E2E validation set is 0.93.

Classifier-based parser q_ϕ It is perhaps too optimistic to believe we can construct reasonable rules in all cases. Rule creation quickly becomes tedious and for more complex MRs this would become a bottleneck. To address these concerns, we also study the feasibility of using learned classifiers to predict the presence and value of the attributes. For each attribute in the E2E dataset, we trained a separate convolutional neural network (CNN) classifier to predict the correct attribute value (or *n/a* if the attribute is not present). The CNN architecture follows that of Kim (2014) and is trained with gradient descent on the original training data. See Appendix C for full architecture and training details. The average E2E validation F-score is 0.94.

5 Self-Training Methodology

Our approach to self-training is relatively straightforward and invariant to the choices of whether or not to use delexicalization, and rule vs. classifier based parser. There are minor differences depending on the dataset and we elaborate on those below. There are three main steps to our self-training approach. Starting with an initially empty augmented dataset $\mathcal{A}$, we

1. Train a base generator model p_0 on the original training data $\mathcal{D}$.
2. Repeat many times:
   (a) Sample a random MR $x \sim \mathcal{X}$.
   (b) Sample K utterances $\tilde{y}^{(i)} \sim p_0(\cdot|x, \epsilon)$
   (c) Parse MR, $\tilde{x}^{(i)} = q(\tilde{y}^{(i)})$, discarding any samples with invalid parses, and adding the survivors to $\mathcal{A}$.
3. Train a new generator p_1 on the combined dataset $\mathcal{D} \cup \mathcal{A}$.

Steps 1 and 3 are identical, the generators p_0 and p_1 have the same architecture and training setup, ony the dataset, $\mathcal{D}$ vs. $\mathcal{D} \cup \mathcal{A}$, is different. We now discuss step 2 in detail.

Step 2: E2E Dataset To sample a novel MR with S attributes, we sample a combination of $S - 1$ attributes uniformly at random (always appending the *name* attribute since every MR contains it). We then sample attribute values for each slot inversely proportional to their empirical frequency in the training set so as to increase the likelihood of creating a novel or under-represented MR.

After obtaining such a sample x we then perform noise injection sampling, generating 200 samples $\tilde{y}^{(i)} \sim p_0(\cdot|x, \epsilon^{(i)})$ in parallel and discarding all but the top 20 samples by average log likelihood according to p_0. We also discard any utterances that have previously been generated.

We then apply the parser to the sampled utterances, to obtain its predicted MR, $\tilde{x}^{(i)} = q(\tilde{y}^{(i)})$. If using the rule based parser $q_{\mathfrak{R}}$ and $\tilde{x} = \emptyset$, i.e. the utterance does not have a valid parse, we discard it. Similarly, when using the classifier based parser, q_ϕ, if any attribute value is predicted with less than 50% probability we discard it. All surviving $(\tilde{x}^{(i)}, \tilde{y}^{(i)})$ pairs are added to $\mathcal{A}$. We repeat this process 25,000 times for each valid MR size S. See Table 8 for statistics on the total sample sizes after filtering.

Step 2: Laptops and TVs On the Laptops and TVs dataset, for each DA and legal number of attributes S we draw S random attributes (modulo any required attributes like *Name*; not all DAs require it).[5]

We then perform noise injection sampling, generating 200 samples $\tilde{y}^{(i)} \sim p_0(\cdot|x, \epsilon^{(i)})$ under the same settings as the E2E dataset. We repeat this process 25,000 times for each DA and DA size. We obtain 373,468 and 33,478 additional samples for the Laptops and TVs datasets respectively.

6 Experiments

6.1 E2E Self-Training

We train base generators p_0 on the original training data $\mathcal{D}$, with and without delexicalizing the *Name* and *Near* attributes. We train for 500 epochs with gradient descent. We use a batch size of 128, with a learning rate of 0.25, weight decay penalty of 0.0001, and a dropout probability of 0.25. We select the best model iteration using validation set BLEU score[6].

Using the self-training method outlined in section 5, we create augmented datasets using either $q_{\mathfrak{R}}$ or q_ϕ, which we refer to as $\mathcal{A}_{q_{\mathfrak{R}}}$ and $\mathcal{A}_{q_\phi}$ respectively (q_ϕ is only in the delexicalized setting).

For both $\mathcal{D} \cup \mathcal{A}_{q_{\mathfrak{R}}}$ and $\mathcal{D} \cup \mathcal{A}_{q_\phi}$ we train new generators p_1 using the same training setting as above (although we terminate training after 50 epochs because the models converge much faster with the additional data).

Results Table 2 shows the automatic quality measurements on the E2E test set using BLEU, ROUGE-L, and METEOR. We show results for both greedy and beam decoding with beam size 8 under p_0 and p_1 models. We compare our models to the best sequence-to-sequence DNN model, Slug (Juraska et al., 2018), the best grammar rule based model, DANGNT (Nguyen and Tran, 2018), and the best template based model, TUDA

		Model	BLEU	R.-L	MET.
		Slug	66.19	67.72	44.54
		DANGNT	59.90	66.34	43.46
		TUDA	56.57	66.14	45.29
delex.	p_0	greedy	66.91	68.27	44.95
		beam	**67.13**	**68.91**	45.15
	p_1 $q_{\mathfrak{R}}$	greedy	65.57	67.71	45.56
		beam	66.28	68.08	**45.78**
	q_ϕ	greedy	63.76	67.31	44.94
		beam	64.23	67.54	45.17
lex.	p_0	greedy	60.35	64.51	41.82
		beam	61.81	65.83	42.69
	p_1 $q_{\mathfrak{R}}$	greedy	64.74	68.21	44.46
		beam	64.81	67.83	44.39

Table 2: BLEU, ROUGE-L, and METEOR metrics on the E2E test set. Baseline methods all rely on at least partial delexicalization, puting our lexicalized models at a relative disadvantage.

(Puzikov and Gurevych, 2018), as determined during the shared task evaluation (Dušek et al., 2019).

Surprisingly, p_0 using greedy decoding surpases all of the baseline systems. This is quite shocking as the Slug model ensembles three different sequence-to-sequence models producing 10 outputs each using beam search and reranking based on slot alignment to select the final generation output. The $p_1/q_{\mathfrak{R}}$ model remains competitive with Slug, again even using greedy decoding. The p_1/q_ϕ starts underperforming Slug on BLEU score but remains competitive on ROUGE-L and METEOR again when using greedy decoding. Overall the augmented training data tends to hurt generation quality. In this regard, the added noise of the trained classifier exacerbates things as it reduces quality more than the rule-based filtering.

In the lexicalized setting, p_0 produces lower quality output than the Slug system. However, the augmented training procedure increases the quality of the lexicalized p_1 model which beats Slug on ROUGE-L.

The automatic quality evaluations are somewhat limited, however. To gain more insight into model performance we apply our rule based parser to estimate attribute realization error for all system outputs on the test set, similarly to (Dušek et al., 2019) (e.g., if the MR specifies *food[French]*, we check to make sure the generated utterance says so). The results of this evaluation are shown in

[5] A number of attributes S is "legal" if we observe a DA instance with that many attributes in the original training data.

[6] We use the official shared task script to compute automatic quality metrics on the E2E dataset.

Model				Name	Near	Family Friendly	Area	Customer Rating	Food	Price Range	Eat Type	All
			Slug	0	0	6	1	6	10	35	9	67
			DANGNT	0	0	18	0	0	0	0	58	76
			TUDA	0	0	0	0	0	0	0	0	**0**
delex.	p_0		greedy	0	0	23	23	16	26	27	0	115
			beam	0	0	60	3	9	3	8	0	83
	p_1	$q_{\mathfrak{R}}$	greedy	0	0	0	0	0	0	0	0	**0**
			beam	0	0	0	0	0	0	0	0	**0**
		q_ϕ	greedy	0	0	1	0	8	1	9	0	19
			beam	0	0	0	0	3	0	0	0	3
lex.	p_0		greedy	145	141	14	15	2	14	2	0	333
			beam	155	124	62	0	0	0	0	0	341
	p_1	$q_{\mathfrak{R}}$	greedy	0	0	2	0	0	125	0	0	127
			beam	0	2	0	0	0	119	0	0	121

Table 3: Attribute realization errors on the E2E test set. The Slug model and our delexicalized models delexicalize the NAME and NEAR slots, thus making 0 errors on these attributes. DANGNT and TUDA models perform complete delexicalization.

Table 3. Immediately, it is revealed that p_0 is far worse than the baseline methods making 115 and 83 errors using greedy and beam decoding respectively.

The $p_1/q_{\mathfrak{R}}$ model achieves zero test set errors even when using the greedy decoding. The p_1/q_ϕ model is slightly worse (in agreement with the automatic quality measurements), but its greedy search is still superior to the more sophisticated Slug decoder, achieving 19 total test set errors compared to Slug's 67 errors.

The lexicalized p_0 model has especially high error rates, particularly on the *Name* and *Near* attributes. With augmented data training, the p_1 model reduces these errors to zero when using greedy search and 2 with beam search. Unfortunately, the augmented training is more unstable in the lexicalized setting, as it produces a large spike in *food* attribute errors, although the p_1 models still have lower overall error than p_0.

6.2 Laptops and TVs Self-Training

We perform similar experiments on the Laptops and TVs datasets. We train a separate p_0 model for each dataset for 300 epochs with a learning rate of 0.1 for Laptops and 0.25 for TVs. The weight decay penalty was 0.0001 and dropout probability was 0.25. Best model iteration is determined by validation set BLEU score. As in the E2E experiments, we create an augmented dataset for both the Laptops and TVs dataset using the method outlined in section 5. We then train new generators p_1 on the union of original training data and the augmented dataset.

Results We automatically evaluate our models using the evaluation script of Wen et al. (2016), which computes BLEU scores, as well as slot alignment error rate (since this dataset is almost fully delexicalized, it simply checks for the presence of the correct attribute placeholders according to the MR). We compare again to the Slug model as well as the Semantically Conditioned LSTM (SCLSTM) (Wen et al., 2015) which report state-of-the-art results on these datasets.

The results are more mixed here. Our BLEU scores are about 15 points below the baselines on the Laptops dataset and 20 points below the baselines on the TVs dataset. Upon examing the evaluation script in detail we see that BLEU score is calculated using 5 model outputs which Juraska et al. (2018) and Wen et al. (2016) do. We only produce the 1-best output at test time, perhaps explaining the difference.

Looking through our model outputs we see mostly good utterances, often nearly exactly matching the references. Our models outperform the state of the art models on errors. The best state of the art models make errors by generating sentences that do not match the input representation

	Laptops		TVs	
Model	BLEU	Err.	BLEU	Err.
SCLSTM	51.16	0.79%	**52.65**	2.31%
Slug	**52.38**	1.55%	52.26	1.67%
p_0 beam	37.13	0.72%	32.63	0.72%
p_1 greedy	37.21	**0.13%**	32.43	0.28%
beam	37.19	0.14%	32.59	**0.20%**

Table 4: BLEU and automatic attribute error on the Laptops and TVs datasets.

0.79% and 1.67% of the time on the Laptops and TVs datasets respectively. Our p_1 model reduces that error to only 0.13% and 0.20%.

6.3 Experiment 4: Human Evaluation

E2E Dataset We had two undergraduate students not involved with the research look at 100 random test set utterances for six of our model variants. They were shown both the Slug output and one of our model outputs and asked to select which output was of better linguistic quality and correctness or indicate that they were equally good. We resolved disagreements in favor of the baseline, i.e. if any annotator thought the baseline was better we considered it so. If an annotator marked one of our systems as better and the other marked it as equal, we considered it equal to the baseline. Inter-annotator agreement was high, with 92% agreement on correctness and 88% agreement on quality.

Table 5 shows the results of the evaluation. We find that the p_1 model outputs are indistinguishable from the Slug model in terms of linguistic quality, regardless of the setting. In terms of correctness, the lexicalized p_1 model is as good as or better than the Slug model 98% of the time. When using the delexicalized models, we don't even need beam search. The delexicalized p_1 greedy decoder is as good as or better than Slug 100% of the time.

Laptops Dataset We had the same annotators look at 100 random *Inform* DAs from the Laptops test set since they are the majority DA type and we could use the same annotator guidelines from the E2E experiment. We do not have access to the Slug or SCLSTM outputs on this dataset, so we compared to one of the two test set reference sentences (picking at random) vs. the $p_1/q_{\mathfrak{R}}$ with greedy decoding. Table 6 shows the results. Despite the low BLEU scores, we find our outputs to be of comparable quality to references 91% of the time. Moreover, they are equally as correct as the human references 100% of the time. Annotators agreed 99% and 87% of the time on correctness and quality respectively.

	Correct.			Quality		
Model	$>$	$=$	$<$	$>$	$=$	$<$
delex. p_0 b	7	89	4	1	96	3
delex. p_1 $q_{\mathfrak{R}}$ g	7	93	0	0	100	0
delex. p_1 $q_{\mathfrak{R}}$ b	7	93	0	0	100	0
delex. p_1 q_ϕ g	5	95	0	0	100	0
delex. p_1 q_ϕ b	8	92	0	0	100	0
lex. p_1 $q_{\mathfrak{R}}$ g	8	90	2	0	100	0

Table 5: Human correctness and quality judgments (%). Comparisons are better than ($>$), equal to ($=$), and worse than ($<$) the baseline Slug model. (g) and (b) indicate greedy and beam decoding respectively.

	Correct.			Quality		
Model	$>$	$=$	$<$	$>$	$=$	$<$
delex. p_1 $q_{\mathfrak{R}}$ g	0	100	0	2	91	7

Table 6: Human correctness and quality judgments (%). Comparisons are better than ($>$), equal to ($=$), and worse than ($<$) the test set references.

7 Sample Analysis and Discussion

We hypothesize that self-training improves the correctness of outputs by sacrificing some expressivity of the model. For example, p_1 BLEU scores on the E2E dataset drop by at least 0.8 as compared to p_0 with beam search. We see a similar pattern on the TVs dataset. Self-training increases automatic metrics in the lexicalized setting, but this could be attributable to reductions in *Name* and *Near* realization errors, which are orthogonal to the syntactic diversity of generation.

To better quantify these effects we report the average length in words, average number of sentences, and average revised Developmental Level (D-Level) score according to the D-Level analyser (Lu, 2009). The D-Level analyser automatically categorizes the syntactic complexities of an utterance into one of eight categories, with eight being the most complex, based on the revised Developmental Level scale (Rosenberg and Abbeduto, 1987; Covington et al., 2006).

Model	Words	Sents	MDL
Human Refs.	24.06	1.76	**2.25**
Slug	24.20	1.86	1.39
lex. p_0 greedy	25.73	2.18	**1.84**
lex. p_0 beam	26.00	2.20	1.50
lex. p_1 $q_{\mathfrak{R}}$ greedy	26.01	2.20	1.39
lex. p_1 $q_{\mathfrak{R}}$ beam	26.04	2.17	1.45
delex. p_0 greedy	24.83	2.10	1.79
delex. p_0 beam	24.51	2.03	1.48
delex. p_1 $q_{\mathfrak{R}}$ greedy	26.50	2.29	1.74
delex. p_1 $q_{\mathfrak{R}}$ beam	26.46	2.28	1.74
delex. p_1 q_ϕ greedy	25.33	1.76	1.77
delex. p_1 q_ϕ beam	25.49	1.75	**1.87**

Table 7: Words/sentences per utterance and mean D-Level score of model outputs on the E2E dataset.

$\mathcal{A}$	Size	Words	Sents	MDL
delex. $\mathcal{A}_{q_\phi}$	384,436	22.5	2.0	1.77
delex. $\mathcal{A}_{q_{\mathfrak{R}}}$	501,909	22.7	2.1	1.76
lex. $\mathcal{A}_{q_{\mathfrak{R}}}$	1,591,778	23.2	2.1	1.69

Table 8: E2E augmented dataset statistics: total utterances, words per utterance, sentences per utterance, and mean D-Level score.

Table 8 shows the statistics for the E2E test set outputs. In the lexicalized setting, the mean D-level results support our hypothesis; syntactic complexity of test set outputs decreases from p_0 to p_1. In the delexicalized setting this is somewhat true; three of the p_1 models have lower mean D-level scores than p_0 with greedy decoding. Curiously, p_1/q_ϕ with beam search has the highest overall syntactic complexity of any our model variants, at odds with our hypothesis. No models are as syntactically complex as the human references, but our models come closest, with a mean D-Level category of 1.87 using the delex. p_1/q_ϕ model with beam decoder.

We also see that $p_1/q_{\mathfrak{R}}$ models are over two sentences in length on average while the human references are under two sentences, suggesting they are more often falling back to simple but reliable ways to realize attributes (e.g., appending "It is a family-friendly venue.").

That our simple models with greedy search and no semantic control mechanisms can perform as reliably as more sophisticated models suggest that in standard training regimes we are often not fully learning from all information available in the training data. Via sampling we can uncover novel recombinations of utterances that are only implied by the provided references. The gains of self-training also suggest that additional research into active learning for this task might bear fruit.

One curious observation about the self-training procedure is that it leads to a convergence in output complexity of greedy and beam decoding. The differences between mean D-level score on the p_0 models is 0.34 and 0.31 in the lexicalized and delexicalized settings respectively. This shrinks to 0.0 and 0.1 in the delexicalized p_1 settings and 0.06 for lexicalized p_1, suggesting that the model probability distributions are sharpening around a smaller set of output structures.

8 Related Work

Neural encoder-decoder models, are a popular choice for dialog generation (Mei et al., 2015; Dušek and Jurčíček, 2016; Chen et al., 2018; Juraska et al., 2018; Elder et al., 2018). However, the quality can vary significantly, with relatively similar architectures yielding both poor and competitve performance (Dušek et al., 2019). All of the cited work on the E2E or Laptops and TVs datasets uses beam search to achieve competitive performance. In addition, they often employ reranking to ensure that all attributes are realized (Dušek and Jurčíček, 2016; Juraska et al., 2018; Wen et al., 2015). (Elder et al., 2018) employ pointer generators to directly copy attribute values, while also using coverage penalties on the attention weights to ensure that all attribute slots are attended to. Unlike these approaches, we do not require beam search, reranking, or other specialized attention mechanisms or loss functions to obtain low error rates. Instead we use data-augmentation to obtain a more reliable but simpler model.

Data augmentation has also been used by prior neural generation models. Juraska et al. (2018) breaks multi-sentence utterances into separate training instances. They also try training on more complex sentences alone but this model was less reliably able to realize all attributes correctly. They also do not generate new utterance/MR pairs for training as we do.

Our method is in some ways similar to the reconstructor setting of Shen et al. (2019), where a base speaker model S_0 produces utterances and a listener model L reconstructs the input MR. In the framework of rational speech acts (RSA) (Monroe and Potts, 2015), a rational speaker model is obtained by composing the base speaker and listener, i.e. $S_1(y|x) = L(x|y) \cdot S_0(y|x)$. While we do not directly compose our parser q and p_0, the p_1 model is learning from the composition of the two. The theoretical commitments of RSA are somewhat orthogonal to our approach. It would be interesing to combine both methods, by incorporating self-training into the RSA framework.

9 Conclusion

We present a novel self-training methodology for learning DNN-based dialogue generation models using noise injection sampling and a MR parser. Even with relatively simple architectures and greedy decoding we are able to match the performance of state-of-the-art baselines on automatic measures of quality while also achieving superior semantic correctness. These findings hold under a human evaluation as well. On automatic measures of syntactic complexity we also find our approach is closer to matching human authored references than prior work. In future work, we intend to explore methods of self-training that futher improve syntactic diversity.

10 Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable feedback, as well as Thomas Effland, Katy Ilonka Gero, and Christopher Hidey for their comments during the drafting of this paper.

This research is based upon work supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via contract #FA8650-17-C-9117. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Mingje Chen, Gerasimos Lampouras, and Andreas Vlachos. 2018. Sheffield at e2e: structured prediction approaches to end-to-end language generation. *E2E NLG Challenge System Descriptions*.

Kyunghyun Cho. 2016. Noisy parallel approximate decoding for conditional recurrent language model. *arXiv preprint arXiv:1605.03835*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Michael A Covington, Congzhou He, Cati Brown, Lorina Naci, and John Brown. 2006. How complex is that sentence? a proposed revision of the rosenberg and abbeduto d-level scale.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2019. Evaluating the state-of-the-art of end-to-end natural language generation: The e2e nlg challenge. *arXiv preprint arXiv:1901.07931*.

Henry Elder, Sebastian Gehrmann, Alexander OConnor, and Qun Liu. 2018. E2e nlg challenge submission: Towards controllable generation of diverse natural language. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 457–462.

Juraj Juraska, Panagiotis Karagiannis, Kevin K Bowden, and Marilyn A Walker. 2018. Slug2slug: A deep ensemble model with slot alignment for sequence-to-sequence natural language generation.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Xiaofei Lu. 2009. Automatic measurement of syntactic complexity in child language acquisition. *International Journal of Corpus Linguistics*, 14(1):3–28.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2015. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

Will Monroe and Christopher Potts. 2015. Learning in the rational speech acts model. *arXiv preprint arXiv:1510.06807*.

Dang Tuan Nguyen and Trung Tran. 2018. Structure-based generation system for e2e nlg challenge. *E2E NLG Challenge System Descriptions*.

Yevgeniy Puzikov and Iryna Gurevych. 2018. E2e nlg challenge: Neural models vs. templates. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 463–471.

Sheldon Rosenberg and Leonard Abbeduto. 1987. Indicators of linguistic competence in the peer group conversational behavior of mildly retarded adults 1. *Applied Psycholinguistics*, 8(1):19–32.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. *arXiv preprint arXiv:1904.01301*.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain neural network language generation for spoken dialogue systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 120–129.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

Compare DA

$$\mathit{Compare}\left(\begin{array}{ll} \mathit{name[Atlas\ 89]} & \mathit{name[Eurus\ 93]} \\ \mathit{isForBusiness[yes]} & \mathit{isForBusiness[no]} \\ \mathit{weight[2.3]} & \mathit{weight[1.12]} \end{array}\right)$$

InformCount DA

$$\mathit{InformCount}\left(\begin{array}{l} \mathit{count[40]} \\ \mathit{family[don't\ care]} \\ \mathit{batteryRating[excellent]} \end{array}\right)$$

	Compare Input	InformCount Input
$x_1 =$	compare	inform_count
$x_2 =$	name	count
$x_3 =$	is_for_biz_yes	family_dont_care
$x_4 =$	weight	batteryrating
$x_5 =$	name	—
$x_6 =$	is_for_biz_no	—
$x_7 =$	weight	—

Compare Utterance

weighing WEIGHT$_1$ kg for business computing the NAME$_1$ is compared to the NAME$_2$ which weighs WEIGHT$_2$ kg and is not for business computing . which one do you like

InformCount Utterance

there are COUNT laptop -s with an BATTERYRATING battery rating if you do not care about the product family.

Figure 2: Example MR, input representations, and utterances for the Laptops and TVs datasets.

Inform DA

$$\mathit{Inform}\left(\begin{array}{l} \mathit{name[The\ Mill]} \\ \mathit{near[Avalon]} \\ \mathit{food[Italian]} \end{array}\right)$$

	Input Lexicalized	Input Delexicalized
x_1	= eat_type_n/a	eat_type_n/a
x_2	= near_avalon	near_present
x_3	= area_n/a	area_n/a
x_4	= fam._friend._n/a	fam._friend._n/a
x_5	= cust._rating_n/a	cust._rating_n/a
x_6	= price_range_n/a	price_range_n/a
x_7	= food_Italian	food_Italian
x_8	= name_the_mill	—

Lexicalized Utterance

the mill serves up italian food near avalon .

Delexicalized Utterance

NAME serves up italian food near NEAR .

Figure 3: Example MR, input representation, and utterance for the E2E dataset.

A Additional Preprocessing Details

MR attributes can be one of two types, dictionary attributes, e.g. *Name* or *CustomerRating*, where the value for the attribute comes from a closed set of valid slot fillers, and binary attributes, e.g. *familyFriendly* or *hasUsbPort*, which can have values *yes* or *no*. Additionally, on the Laptops and TVs datasets, attributes can also have a distinguished *don't care* value, e.g. the MR *InformCount*$\big($*count[40], priceRange[don't care]*$\big)$ could be realized as "There are 40 laptops available if you do not care about the price range."

When using the delexicalied model, we do not represent the *Name* attribute in the input x, since every valid E2E MR contains this attribute, i.e. the model learns on its own to always generate the *NAME* token to be replaced at test time.

Certain attributes were inconsistently realized in the reference utterances. E.g., utterances would frequently refer to locations as restaurants even when the *eatType* attribute was not present so we ammended the MR to have the attribute *eatType[restaurant]* in those cases. Additionally, *priceRange[cheap]* and *priceRange[less than £20]* would be interchanged; *cheap* utterances that mentioned numerical amounts were remapped to *less than £20* and visa-versa. Similar corrections were made for the *customerRating* attribute. Note, these changes were only done on the training and validation set. We do not modify the test set at all.

B Model Input Representation

We show an example input sequence x for the E2E dataset in Figure 3 and for the Laptops and TVs datasets in Figure 2.

C CNN Classifier Details

We use a separate CNN classifier for each attribute to predict the corresponding value (or *n/a*) from an utterance y. We first look up the tokens in y in an embedding matrix E to obtain a matrix $W \in \mathbb{R}^{N \times D}$ where $D = 50$ is the embedding dimension.

We then apply a series of unigram, bigram, and trigram convolutional filters each with 50 output features. After concatenating and max-pooling over the sequence dimension, and applying a ReLU activation, we obtain a hidden layer in $\mathbb{R}^{150}$. We then apply another fully-connected layer with ReLU activation which down projects the hidden layer to $\mathbf{R}^{50}$. Finaly we apply the final softmax layer to predict the class label.

During training we apply dropout (with drop rate 0.25) to the embedding layer, convolutional filter outputs, and hidden layers. We train for 30 epochs with gradient descent using a learning rate of 0.25 and weight decay penalty of 0.0001, using validation set F1 as our model selection criterion.

We treat the utterance as a

D Augmented Dataset Samples

Samples and their parsed MR for the E2E datsets are shown in Table 9, Table 10, and Table 11.

Inform(*eatType*[pub] *food*[Italian] *name*[NAME] *priceRange*[high])
NAME is a pub that serves italian food in the high price range .

Inform(*area*[city centre] *customerRating*[5 out of 5] *eatType*[coffee shop] *familyFriendly*[no] *food*[Indian] *name*[NAME])
the NAME is a coffee shop that serves indian food . it is located in the city centre and has a customer rating of 5 out of 5 . it is not kid friendly .

Inform(*customerRating*[3 out of 5] *eatType*[pub] *food*[English] *name*[NAME] *near*[NEAR])
the NAME is a pub that serves english food and is located near the NEAR . it has a customer rating of 3 out of 5 .

Inform(*area*[city centre] *customerRating*[3 out of 5] *eatType*[pub] *familyFriendly*[yes] *food*[Fast food] *name*[NAME] *priceRange*[less than £20])
the NAME is a pub in the city centre that serves fast food for less than £20 . it has a customer rating of 3 out of 5 and is family - friendly .

Inform(*customerRating*[high] *eatType*[coffee shop] *food*[French] *name*[NAME] *near*[NEAR] *priceRange*[cheap])
NAME is a cheap coffee shop near NEAR that serves french food . it has a high customer rating .

Inform(*area*[city centre] *customerRating*[5 out of 5] *eatType*[pub] *familyFriendly*[yes] *food*[Chinese] *name*[NAME] *priceRange*[cheap])
the NAME is a family - friendly pub in the city centre . it serves cheap chinese food and has a customer rating of 5 out of 5 .

Inform(*customerRating*[high] *familyFriendly*[yes] *food*[Indian] *name*[NAME] *priceRange*[high])
the NAME serves high priced indian food . it has a high customer rating and is child friendly .

Inform(*customerRating*[3 out of 5] *food*[Japanese] *name*[NAME] *near*[NEAR] *priceRange*[less than £20])
NAME serves japanese food for less than £20 . it is located near NEAR and has a customer rating of 3 out of 5 .

Table 9: Example E2E samples obtained using noise injection sampling with delex. p_0 and $q_{\mathfrak{R}}$ to parse the MR.

Inform(*customerRating*[3 out of 5] *familyFriendly*[no] *food*[English] *name*[NAME] *priceRange*[£20-25]) the NAME serves english food in the £20 - £25 price range . it is not kid friendly and has a customer rating of 3 out of 5
Inform(*area*[riverside] *customerRating*[3 out of 5] *eatType*[restaurant] *familyFriendly*[yes] *food*[Chinese] *name*[NAME] *near*[PRESENT] *priceRange*[high]) NAME is a family friendly restaurant serving chinese food in the riverside area near NEAR . it has a high price range and a customer rating of 3 out of 5
Inform(*area*[city centre] *eatType*[restaurant] *food*[English] *name*[NAME] *priceRange*[less than £20]) NAME is a restaurant providing english food in the less than £20 price range . it is located in the city centre
Inform(*customerRating*[3 out of 5] *food*[Italian] *name*[NAME] *near*[PRESENT] *priceRange*[£20-25]) NAME has a price range of £20 - 25 . it has a customer rating of 3 out of 5 and serves italian food . it is near the NEAR
Inform(*area*[riverside] *customerRating*[3 out of 5] *food*[Chinese] *name*[NAME] *priceRange*[moderate]) the NAME serves chinese food in the riverside area . it has a moderate price range and a customer rating of 3 out of 5
Inform(*area*[riverside] *customerRating*[low] *familyFriendly*[yes] *food*[English] *name*[NAME]) the NAME serves english food in the riverside area . it has a low customer rating and is kid friendly
Inform(*customerRating*[low] *eatType*[coffee shop] *familyFriendly*[yes] *food*[Japanese] *name*[NAME]) the NAME is a family friendly japanese coffee shop with a low customer rating
Inform(*customerRating*[1 out of 5] *familyFriendly*[no] *food*[Japanese] *name*[NAME] *near*[PRESENT]) NAME serves japanese food near NEAR . it is not kid friendly and has a customer rating of 1 out of 5

Table 10: Example E2E samples obtained using noise injection sampling with delex. p_0 and q_ϕ to parse the MR.

Inform(*area*[riverside] *customerRating*[5 out of 5] *eatType*[restaurant] *food*[French] *name*[The Eagle] *near*[The Sorrento] *priceRange*[less than £20])
the eagle is a french restaurant with a 5 out of 5 customer rating and a price range of less than £20 . it is located in the riverside area near the sorrento .

Inform(*area*[city centre] *eatType*[coffee shop] *familyFriendly*[no] *food*[English] *name*[The Wrestlers] *near*[Raja Indian Cuisine])
the wrestlers is a coffee shop that serves english food . it is located in the city centre near raja indian cuisine . it is not child friendly .

Inform(*area*[riverside] *name*[Taste of Cambridge] *priceRange*[cheap])
taste of cambridge is located in the riverside area . it is cheap .

Inform(*customerRating*[3 out of 5] *eatType*[restaurant] *familyFriendly*[yes] *name*[Zizzi])
there is a kid friendly restaurant called zizzi . it has a customer rating of 3 out of 5 .

Inform(*area*[city centre] *eatType*[pub] *name*[The Cambridge Blue] *near*[Yippee Noodle Bar] *priceRange*[high])
the cambridge blue is a pub in the high price range . it is located in the city centre near the yippee noodle bar .

Inform(*area*[riverside] *customerRating*[average] *name*[The Phoenix])
the phoenix is located in the riverside area near the riverside . it has an average customer rating .

Inform(*customerRating*[average] *familyFriendly*[yes] *food*[Indian] *name*[Loch Fyne])
loch fyne provides indian food . it is family friendly and has an average customer rating .

Inform(*area*[riverside] *customerRating*[1 out of 5] *food*[Italian] *name*[The Phoenix] *near*[The Six Bells])
the phoenix is located in the riverside area near the city centre , near the six bells . it serves italian food and has a customer rating of 1 out of 5 .

Table 11: Example E2E samples obtained using noise injection sampling with lex. p_0 and $q_{\mathfrak{R}}$ to parse the MR.