Controllable Meaning Representation to Text Generation: Linearization and Data Augmentation Strategies

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

We study the degree to which neural sequence-to-sequence models exhibit fine-grained controllability when performing natural language generation from a meaning representation. Using two task-oriented dialogue generation benchmarks, we systematically compare the effect of four input linearization strategies on controllability and faithfulness. Additionally, we evaluate how a phrase-based data augmentation method can improve performance. We find that properly aligning input sequences during training leads to highly controllable generation, both when training from scratch or when fine-tuning a larger pre-trained model. Data augmentation further improves control on difficult, randomly generated utterance plans.

MR/Utterance Pair

REQUEST EXPLANATION
genres = [
  "role-playing",$^{(1)}$
  "hack-and-slash",$^{(2)}$]
ESRB = "M (Mature)"$^{(3)}$
rating = "good"$^{(4)}$

What is it about M rated^{3} hack-and-slash^{2} RPGs^{1} that makes you enjoy them?^{4}

Figure 1: Example MR for REQUEST EXPLANATION dialogue act (left) and utterance (right) pair from the ViGGO dataset. Superscripts indicate which attribute-values correspond to which utterance subspans.

1 Introduction

In this work, we study the degree to which neural sequence-to-sequence (S2S) models exhibit fine-grained controllability when performing natural language generation (NLG) from a meaning representation (MR). In particular, we focus on an S2S approach that respects the realization ordering constraints of a given utterance plan; such a model can generate utterances whose phrases follow the order of the provided plan.

In non-neural NLG, fine-grained control for planning sentence structure has received extensive study under the names *sentence* or *micro-planning* (Reiter and Dale, 2000; Walker et al., 2001; Stone et al., 2003). Contemporary practice, however, eschews modeling at this granularity, instead preferring to train an S2S model to directly map an input MR to a natural language utterance, with the utterance plan determined implicitly by the model which is learned from the training data (Dušek et al., 2020).

We argue that robust and fine grained control in an S2S model is desirable because it enables neural implementations of various psycho-linguistic theories of discourse (e.g., Centering Theory (Grosz et al., 1995), or Accessibility Theory (Ariel, 2001)). This could, in turn, encourage the validation and/or refinement of additional psychologically plausible models of language production.

In this paper, we study controllability in the context of task-oriented dialogue generation (Mairesse et al., 2010; Wen et al., 2015), where the input to the NLG model is an MR consisting of a dialogue act (i.e. a communicative goal) such as to REQUEST EXPLANATION, and an unordered set of attribute-value pairs defining the semantics of the intended utterance (see Figure 1 for an example).

The NLG model is expected to produce an utterance that adequately and faithfully communicates the MR. In the S2S paradigm, the MR must be "linearized" (i.e. represented as a linear sequence of tokens corresponding to the dialogue act and attribute-value pairs) before being presented to the S2S encoder. We explore several linearization strategies and measure their effectiveness for controlling phrase order, as well as their effect on model faithfulness (i.e., the semantic correctness of generated utterances).

Of particular note, *alignment training* (i.e. at training time, linearizing the attribute-value pairs according to the order in which they are realized by their corresponding reference utterance) produces highly controllable S2S models. While we are not the first to observe this (c.f., Nayak et al. (2017)), we study this behavior extensively. We refer to an ordered sequence of attribute-value pairs $x_1, x_2, \ldots, x_n$ as an *utterance plan*, and evaluate models on their ability to follow such plans given by either another model, a human, or, most difficultly, from random permutation.

Additionally, we experiment with a data augmentation method, where we create fragmentary MR/utterance pairs obtained from the constituent phrases of the original training data. We find that this data augmentation results in reduced semantic error rates and increases the ability of a model to follow an arbitrary utterance plan.

We summarize our contributions as follows. (1) We show that **alignment training produces highly controllable language generation models**, especially when following a model created utterance plan. (2) We demonstrate that **phrase-based data augmentation improves the robustness of the control** even on arbitrary and difficult to follow utterance plans. (3) We conclude with a human evaluation that shows that **phrase-based data augmentation training can increase the robustness of control without hurting fluency.**[1]

2 Methods

In an MR-to-text task, we are given as input an MR $\mu \in \mathcal{M}$ from which to generate an appropriate natural language utterance $\mathbf{y} \in \mathcal{Y}$, where μ consists of a dialogue act that characterizes the communicative goal of the utterance and an unordered and variably sized set of attribute-value pairs. Attributes are either binary or categorical variables (e.g., *family-friendly*: ["yes", "no"] or *food*: ["Chinese", "English", "French", ...]).[2]

Let each attribute-value pair x and dialogue act a be tokens from a vocabulary $\mathcal{V}$, and define the size of an MR, denoted $|\mu|$, to be the number of attribute-value pairs $x \in \mu$. A linearization strategy $\pi : \mathcal{M} \rightarrow \mathcal{V}^*$ is a mapping of the dialogue act and attribute-value pairs in μ to an ordered sequence, i.e. $\pi(\mu) = [a, x_1, x_2, \ldots, x_{|\mu|}]$. Regardless of the choice of π, the first token in $\pi(\mu)$ is always the dialogue act a.

We experiment with both gated recurrent unit (GRU) (Cho et al., 2014) and Transformer (Vaswani et al., 2017) based S2S model variants to implement a conditional probability model $p(\cdot|\pi(\mu);\theta) : \mathcal{Y} \rightarrow (0,1)$ over utterances. The model parameters, θ, are learned by approximately maximizing the log-likelihood $\mathcal{L}(\theta) = \sum_{(\mu,\mathbf{y})\in\mathcal{D}} \log p(\mathbf{y}|\pi(\mu);\theta)$ on the training set $\mathcal{D}$. Additionally, we experiment with a pretrained S2S Transformer, BART (Lewis et al., 2020), with parameters θ_0 fine-tuned on $\mathcal{L}(\theta_0)$.

2.1 Linearization Strategies

Because of the recurrence in the GRU and position embeddings in the Transformer, it is usually the case that different linearization strategies, i.e. $\pi(\mu) \neq \pi'(\mu)$, will result in different model internal representations and therefore different conditional probability distributions. These differences can be non-trivial, yielding changes in model behavior with respect to faithfulness and control.

We study four linearization strategies, *(i) random*, *(ii) increasing-frequency*, *(iii) fixed-position*, and *(iv) alignment training*, which we describe below. For visual examples of each strategy, see Figure 2. Note that linearization determines the order of the attribute-value pairs presented to the S2S encoder, and **only** in the case of alignment training does it correspond to the order in which the attribute-value pairs are realized in the utterance. When presenting a linearized MR to the model encoder, we always prepend and append distinguished *start* and *stop* tokens respectively.

Random (RND) In the *random* linearization (RND), we randomly order the attribute-value pairs for a given MR. This strategy serves as a baseline for determining if linearization matters at all for faithfulness. RND is similar to token level noise used in denoising auto-encoders (Wang et al., 2019) and might even improve faithfulness. During training, we resample the ordering for each example at every epoch. We do not resample the validation set in order to obtain stable results for model selection.

Increasing Frequency (IF) In the *increasing frequency* linearization (IF), we order the attribute-value pairs by increasing frequency of occurrence in the training data (i.e., $\text{count}(x_i) \leq$

[1] Code, outputs, augmented data, and other materials can be found here: https://github.com/kedz/cmr2text.

[2] There are also list-valued attributes but we treat them as individual attribute-value pairs (i.e. in Figure 1, both *genres* = "role-playing" and *genres* = "hack-and-slash" are in μ).

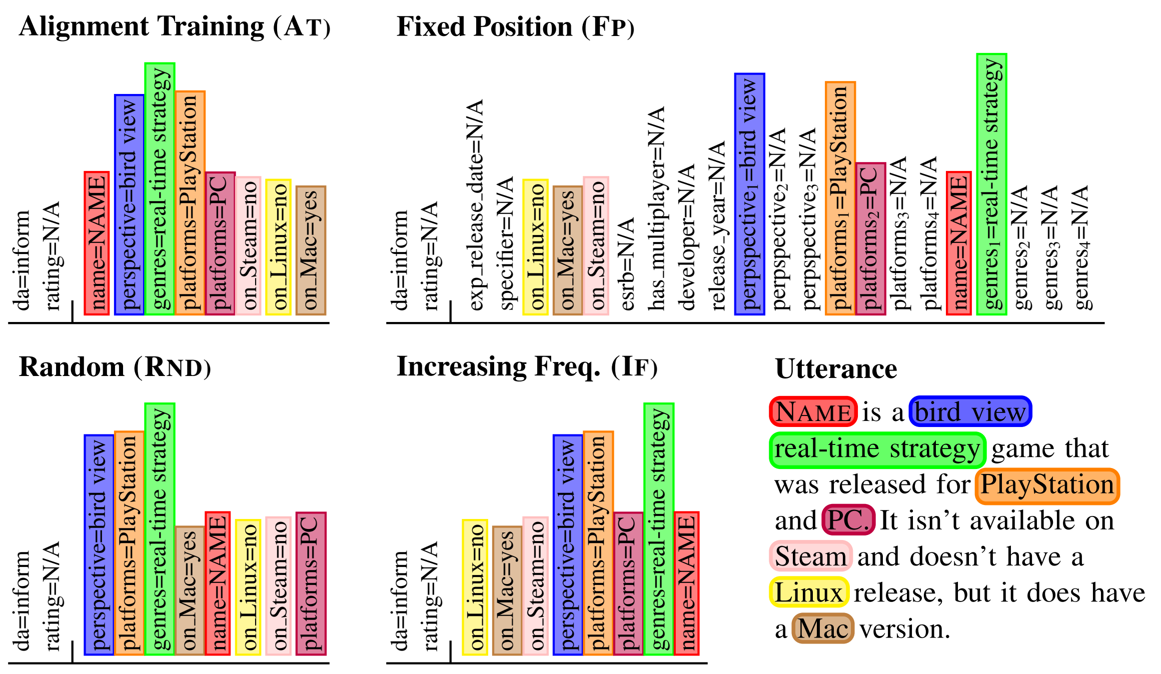

Figure 2: Example MR linearization strategies for the utterance above from the ViGGO training set.

count(x_{i+1})). We hypothesize that placing frequently occurring items in a consistent location may make it easier for p to realize those items correctly, possibly at the expense of rarer items.

Fixed Position (FP) We take consistency one step further and create a fixed ordering of all attributes, *n.b.* not attribute-values, ordering them in increasing frequency of occurrence on the training set (i.e. every instance has the same order of attributes in the encoder input). In this *fixed position* linearization (FP), attributes that are not present in an MR are explicitly represented with an "N/A" value. For list-valued slots, we determine the maximum length list in the training data and create that many repeated slots in the input sequence. This linearization is feasible for datasets with a modest number of unique attributes but may not easily scale to 10s, 100s, or larger attribute vocabularies.

Alignment Training (AT) In the *alignment training* linearization (AT), during training, the order of attribute-value pairs $x_1, x_2, \ldots, x_{|\mu|}$ matches the order in which they are realized in the corresponding training utterance. This is feasible because in the majority of cases, there is a one-to-one mapping of attribute-values and utterance subspans.

We obtain this ordering using a manually constructed set of matching rules to identify which utterance subspans correspond to each attribute-value pair (see §3.1).

$\pi(\mu)$ = [inform, name=Aromi, eat_type=coffee shop, area=city centre]
$\hat{\mathbf{y}}$ = *Aromi is a coffee shop in the city centre.*

$\pi(\mu)$ = [inform, eat_type=coffee shop, name=Aromi, area=city centre]
$\hat{\mathbf{y}}$ = *There is a coffee shop called Aromi in the city centre.*

$\pi(\mu)$ = [inform, eat_type=coffee shop, area=city centre, name=Aromi]
$\hat{\mathbf{y}}$ = *For coffee in the centre of the city, try Aromi.*

Figure 3: Example outputs ($\hat{\mathbf{y}}$) from a controllable model, i.e. a S2S model trained with AT linearization, under different input utterance plans ($\pi(\mu)$).

Crucially, AT stands in contrast to the first three strategies (RND, IF, and FP) which do not have any correspondence between the the order of attribute-value pairs in $\pi(\mu)$ and the order in which they are realized in the corresponding utterance $\mathbf{y}$.

At test time, when there is no reference utterance AT cannot specify a linearization. However, models trained with AT can generate an utterance from an arbitrary utterance plan $x_1, x_2, \ldots, x_{|\mu|}$ provided by an external source, such as an utterance planner model or human reference. See Figure 3 for an example of how an AT-trained model might follow three different plans for the same MR.

2.2 Phrase-based Data Augmentation

We augment the training data with MR/utterance pairs taken from constituent phrases in the original training data. We parse all training utterances and enumerate all constituent phrases governed by NP,

	E2E		Viggo	
	Orig.	Phr.	Orig.	Phr.
# ex.	33,523	443,192	5,103	67,445
Avg. $\|\mu\|$	5.3	1.8	6.8	3.8
Avg. $\|\mathbf{y}\|$	23.6	7.0	24.4	6.7

Table 1: Statistics of original training and phrase data.

VP, ADJP, ADVP, PP, S, or Sbar non-terminals.[3] We then apply the attribute-value matching rules used for AT (see §3.1) to obtain a corresponding MR, keeping the dialogue act of the original utterance. We discard phrases with no realized attributes. See Table 1 for augmented data statistics.

Because we reclassify the MR of phrases using the matching rules, the augmented data includes examples of how to invert binary attributes, e.g. from the phrase "*is not on Mac,*" which denotes *has_mac_release* = "no," we obtain the phrase "*on Mac*" which denotes *has_mac_release* = "yes." When presenting the linearized MR of phrase examples to the model encoder we prepend and append phrase specific *start* and *stop* tokens respectively (e.g., *start-NP* and *stop-NP*) to discourage the model from ever producing an incomplete sentence when generating for a complete MR.

3 Datasets

We run our experiments on two English language, task-oriented dialogue datasets, the E2E Challenge corpus (Novikova et al., 2017) and the ViGGO corpus (Juraska et al., 2019). These datasets provide MR/utterance pairs from the restaurant and video game domains, respectively. Examples from the E2E corpus (33,523 train/1,426 dev/630 test) can have up to eight unique attributes. There is only one dialogue act for the corpus, INFORM. Attribute-values are either binary or categorical valued.

The ViGGO corpus (5,103 train/246 dev/359 test) contains 14 attribute types and nine dialogue acts. In addition to binary and categorical valued attributes, the corpus also features list-valued attributes (see the *genres* attribute in Figure 1) which can have a variable number of values, and an open-class *specifier* attribute (see §A.1 for details).

3.1 MR/Utterance Alignments

The original datasets do not have alignments between individual attribute-value pairs and the subspans of the utterances they occur in, which we need for the AT linearization strategy. We manually developed a list of heuristic pattern matching rules (e.g. *not kid-friendly* $\rightarrow$ *family_friendly* = "no"). For ViGGO, we started from scratch, but for E2E we greatly expanded the rule-set created by Dušek et al. (2019). To ensure the correctness of the rules, we iteratively added new matching rules, ran them on the training and validation sets, and verified that they produced the same MR as was provided in the dataset. This process took one author roughly two weeks to produce approximately 25,000 and 1,500 rules for the E2E and ViGGO datasets respectively. Note that the large number of rules is obtained programmatically, i.e. creating template rules and inserting matching keywords or phrases (e.g., enumerating variants such as *not kid-friendly*, *non kid-friendly*, *not family-friendly*, etc.).

In cases where the matching rules produced different MRs than provided in the original dataset, we manually checked them. In many cases on the E2E dataset and several times on ViGGO, we found the rule to be correct and the MR to be incorrect for the given utterance. In those cases, we used the corrected MRs for training and validation. We do not modify the test sets in any way. Using the matching rules, we can determine alignments between the provided MR and the realized utterances.

For most cases, the attribute-values uniquely correspond to a non-overlapping subspan of the utterance. The *rating* attribute in the ViGGO dataset, however, could have multiple reasonable mappings to the utterance, so we treat it in practice like an addendum to the dialogue act, occurring directly after the dialogue act as part of a "header" section in any MR linearization strategy (see Figure 2 where *rating* = "N/A" occurs after the dialogue act regardless of choice of linearization strategy).

4 Models

4.1 Generation Models

We examine the effects of linearization strategy and data augmentation on a bidirectional GRU with attention (biGRU) and Transformer-based S2S models. Hyperparameters were found using grid-search, selecting the model with best validation BLEU (Papineni et al., 2002) score. We performed a separate grid-search for each architecture-linearization strategy pairing in case there was no one best hyperparameter setting.

[3]We used the Stanford CoreNLP parser v3.9.2.

Additionally, we fine-tune BART (Lewis et al., 2020), a large pretrained Transformer-based S2S model. We stop fine-tuning after validation set cross-entropy stops decreasing.

Complete architecture specification, hyperparameter search space, and validation results for all three models can be found in Appendix A.

Decoding When decoding at test time, we use beam search with a beam size of eight. Beam candidates are ranked by length normalized log likelihood. Similar to Dušek et al. (2019) and Juraska et al. (2019) we rerank the beam output to maximize the F-measure of correctly generated attribute-values using the matching-rules described in §3.1.

For models using the RND linearization, at test time, we sample five random MR orderings and generate beam candidates for each. Reranking is then performed on the union of beam candidates.

4.2 Utterance Planner Model

We experiment with three approaches to creating a test-time utterance plan for the AT trained models. The first is a bigram language model (BGUP) over attribute-value sequences. Attribute-value bigram counts are estimated from the training data (using Lidstone smoothing (Chen and Goodman, 1996) with $\alpha = 10^{-6}$) according to the ordering determined by the matching rules (i.e. the AT ordering).

The second model is a biGRU based S2S model, which we refer to as the neural utterance planner (NUP). We train the NUP to map IF ordered attribute-values to the AT ordering. We grid-search model hyperparameters, selecting the model with highest average Kendall's τ (Kendall, 1938) on the validation set AT orderings. See Appendix B for hyperparameter/model specification details. Unlike the BGUP model, the NUP model also conditions on the dialogue act, so it can learn ordering preferences that differ across dialogue acts.

For both BGUP and NUP, we use beam search (with beam size 32) to generate candidate utterance plans. The beam search is constrained to only generate attribute-value pairs that are given in the supplied MR, and to avoid generating repeated attributes. The search is not allowed to terminate until all attribute-values in the MR are generated. Beam candidates are ranked by log likelihood.

The final ordering we propose is the ORACLE ordering, i.e. the utterance plan implied by the human-authored test-set reference utterances. This plan represents the model performance if it had *a priori* knowledge of the reference utterance plan. When a test example has multiple references, we select the most frequent ordering in the references, breaking ties according to BGUP log-likelihood.

5 Experiments

5.1 Test-Set Evaluation

In our first experiment, we compare performance of the proposed models and linearization strategies on the E2E and ViGGO test sets. For the IF and AT+NUP models we also include variants trained on the union of original training data and phrase-augmented data (see §2.2), which we denote +P.

Evaluation Measures For automatic quality measures, we report BLEU and ROUGE-L (Lin, 2004) scores.[4] Additionally, we use the matching rules to automatically annotate the attribute-value spans of the model generated utterances, and then manually verify/correct them. With the attribute-value annotations in hand we compute the number of missing, wrong, or added attribute-values for each model. From these counts, we compute the semantic error rate (SER) (Dušek et al., 2020) where

$$\text{SER} = \frac{\#missing + \#wrong + \#added}{\#attributes}.$$

On ViGGO, we do not include the *rating* attribute in this evaluation since we consider it part of the dialogue act. Additionally, for AT variants, we report the order accuracy (OA) as the percentage of generated utterances that correctly follow the provided utterance plan. Utterances with wrong or added attribute values are counted as not following the utterance plan. Additional metrics and SER error break downs can be found in Appendix C.

All models are trained five times with different random seeds; we report the mean of all five runs. We report statistical significance using Welch's t-test (Welch, 1947), comparing the score distribution of the five runs from the best linearization strategy against all other strategies at the 0.05 level.

Baselines On the ViGGO dataset we compare to the Transformer baseline of Juraska et al. (2019), which used a beam search of size 10 and heuristic slot reranker (similar to our matching rules).

On the E2E dataset, we report the results of TGen+ (Dušek et al., 2019), an LSTM-based S2S

[4]We use the official E2E evaluation script to compute these numbers.

model, which also uses beam search with a matching rule based reranker to select the most semantically correct utterance and is trained on a cleaned version of the corpus (similar to our approach).

5.2 Random Permutation Stress Test

Differences between an AT model following a utterance planner model and the human oracle are often small so we do not learn much about the limits of controllability of such models, or how they behave in extreme conditions (i.e. on an arbitrary, random utterance plan, not drawn from the training data distribution). In order to perform such an experiment we generate random utterance plans (i.e. permutations of attribute-values) and have the AT models generate utterances for them, which we evaluate with respect to SER and OA (we lack ground truth references with which to evaluate BLEU or ROUGE-L). We generate random permutations of size $3, 4, \ldots, 8$ on the E2E dataset, since there are 8 unique attributes on the E2E dataset. For ViGGO we generate permutations of size $3, 4, \ldots, 10$ (96% of the ViGGO training examples fall within this range). For each size we generated 100 random permutations and all generated plans were given the INFORM dialogue act. In addition to running the AT models on these random permutations, we also compare them to the same model after using the NUP to reorder them into an easier[5] ordering. Example outputs can be found in Appendix D.

5.3 Human Evaluation

In our final experiment, we had human evaluators rank the 100 outputs of the size 5 random permutations for three BART models on both datasets: (i) AT+P model with NUP, (ii) AT+P model, and (iii) AT model. The first model, which uses an utterance planner, is likely to be more natural since it doesn't have to follow the random order, so it serves as a ceiling. The second and third models will try to follow the random permutation ordering, and are more likely to produce unnatural transitions between awkard sequences of attribute-values. Differences between these models will allow us to understand how the phrase-augmented data affects the fluency of the models. The annotators were asked to rank outputs by their naturalness/fluency. Each set was annotated twice by different annotators so we can compute agreement. More details can be found in Appendix E.

[5]Easier in the sense that the NUP re-ordering is closer to the training set distribution of AT utterance plans.

6 Results

AT models accurately follow utterance plans. See Table 2 and Table 3 for results on E2E and ViGGO test sets respectively. The best non-ORACLE results are bolded for each model and results that are not different with statistical significance to the best results are underlined. We see that the AT+NUP strategy consistently receives the lowest semantic error rate and highest order accuracy, regardless of architecture or dataset, suggesting that alleviating the model's decoder of content planning is highly beneficial to avoiding errors. The Transformer AT model is able to consistently achieve virtually zero semantic error on E2E using either the bigram or neural planner model.

We also see that fine-tuned BART is able to learn to follow an utterance plan as well. When following the neural utterance planner, BART is highly competitive with the trained from scratch Transformer on E2E and surpassing it on ViGGO in terms of semantic error rate.

Generally, the AT models had a smaller variance in test-set evaluation measures over the five random initializations as compared to the other strategies. This is reflected in some unusual equivalency classes by statistical significance. For example, on the E2E dataset biGRU models, the AT+NUP+P strategy acheives 0% semantic error and is significantly different than all other linearization strategies **except** the FP strategy even though the absolute difference in score is 6.54%. This is unusual because the AT+NUP+P strategy **is** significantly different from AT+NUP but the absolute difference is only 0.26%. This happens because the variance in test-set results is higher for FP making it harder to show signficance with only five samples.

Transformer-based models are more faithful than biGRU on RND, FP, and IF linearizations. On the ViGGO dataset, BART and Transformer IF achieve 1.86% and 7.50% semantic error rate respectively, while the biGRU IF model has 19.20% semantic error rate. These trends hold for FP and RND, and on the E2E dataset as well. Because there is no sequential correspondence in the input, it is possible that the recurrence in the biGRU makes it difficult to ignore spurious input ordering effects. Additionally, we see that RND does offer some

[6]Since their model does not realize *specifier* attributes, we do not include them in SER calculation. When including them, their model achieves 2.6% SER.

	Model	B↑	R↑	SER↓	OA↑
	TGen+ (Dušek et al., 2019)	66.0	67.6	0.03	—
biGRU	Rnd	**66.8**	68.3	2.64	—
	Fp	63.4	65.6	6.54	—
	If	59.2	62.7	12.64	—
	If+p	65.8	68.1	0.24	—
	At+BgUP	66.4	68.3	0.26	98.2
	At+NUP	66.3	68.9	0.26	98.3
	At+NUP+p	66.5	**69.1**	**0.00**	**100.0**
	At Oracle	69.8	77.3	0.84	94.3
Transformer	Rnd	**67.4**	68.2	1.06	—
	Fp	**67.4**	68.7	3.10	—
	If	67.1	68.1	0.66	—
	If+p	66.8	68.3	0.28	—
	At+BgUP	66.8	68.4	**0.00**	99.9
	At+NUP	67.0	69.0	**0.00**	**100.0**
	At+NUP+p	66.7	**69.1**	**0.00**	**100.0**
	At Oracle	69.3	77.0	0.76	95.0
BART	Rnd	66.5	68.3	0.14	—
	Fp	65.5	67.2	0.16	—
	If	65.6	67.4	0.18	—
	If+p	65.9	68.2	0.30	—
	At+BgUP	66.2	68.7	0.20	98.6
	At+NUP	**66.6**	69.2	0.20	98.6
	At+NUP+p	66.3	**69.3**	**0.00**	**100.0**
	At Oracle	68.3	77.1	0.70	95.3

Table 2: E2E test set (B) Bleu, (R) Rouge-L, SER, and OA. All numbers are percents.

	Model	B↑	R↑	SER↓	OA↑
	Transformer (Juraska et al., 2019)	52.1	63.8	1.60[6]	—
biGRU	Rnd	50.2	61.6	12.56	—
	Fp	50.2	61.0	17.12	—
	If	50.2	61.3	19.20	—
	If+p	49.5	61.6	12.46	—
	At+BgUP	48.5	58.5	3.40	89.8
	At+NUP	51.8	62.6	**1.58**	93.7
	At+NUP+p	**52.4**	**62.7**	1.62	**94.3**
	At Oracle	54.1	65.5	2.42	92.2
Transformer	Rnd	52.0	62.9	9.62	—
	Fp	**52.6**	63.0	8.70	—
	If	52.3	62.6	7.50	—
	If+p	52.3	**63.1**	4.24	—
	At+BgUP	48.7	59.2	4.68	79.1
	At+NUP	51.6	62.4	2.70	88.3
	At+NUP+p	51.1	62.0	**2.28**	**89.8**
	At Oracle	53.2	65.0	4.08	83.0
BART	Rnd	43.7	55.1	1.50	—
	Fp	47.0	58.9	1.68	—
	If	43.1	54.4	1.86	—
	If+p	**49.1**	**59.7**	1.78	—
	At+BgUP	43.8	54.0	0.52	**98.3**
	At+NUP	45.5	57.6	0.54	98.2
	At+NUP+p	48.5	59.2	**0.46**	98.1
	At Oracle	47.1	60.4	0.82	97.2

Table 3: ViGGO test set (B) Bleu, (R) Rouge-L, SER, and OA. All numbers are percents.

benefits of denoising; Rnd models have lower semantic error rate than If models in 3 of 6 cases and Fp models in 5 out of 6 cases.

Model based plans are easier to follow than human reference plans. On E2E, there is very little difference in semantic error rate when following either the bigram-based utterance planner, BgUP, or neural utterance planner, NUP. This is also true of the ViGGO BART models as well. In the small data (i.e. ViGGO) setting, biGRU and Transformer models achieve better semantic error rate when following the neural utterance planner. In most cases, neural utterance planner models have slightly higher Bleu and Rouge-L than the bigram utterance planner, suggesting the neural planner produces utterance plans closer to the reference orderings. The neural and bigram planner models have slightly lower semantic error rate than when following the Oracle utterance plans. This suggests that the models are producing orders more commonly seen in the training data, similar to how neural language generators frequently learn the least interesting, lowest entropy responses (Serban et al., 2016). On the other hand, when given the Oracle orderings, models achieve much higher word overlap with the reference, e.g. achieving an E2E Rouge-L ≥ 77.

Phrase-training reduces SER. We see that phrase data improves semantic error rate in 8 out of 12 cases, with the largest gains coming from the biGRU If model. Where the base semantic error rate was higher, phrase training has a more noticeable effect. After phrase training, all E2E models are operating at near zero semantic error rate and almost perfectly following the neural utterance planner. Model performance on ViGGO is more varied, with phrase training slighting hurting

	E2E		ViGGo	
Model	SER↓	OA↑	SER↓	OA↑
biGRU	1.14	94.44	13.58	46.72
+P	0.54	97.34	14.46	49.26
+NUP	0.22	98.72	9.62	62.04
+NUP+P	**0.02**	**99.86**	**8.98**	**64.50**
Transformer	0.78	95.20	28.34	18.70
+P	0.40	98.10	25.72	18.10
+NUP	0.08	99.64	24.18	31.34
+NUP+P	**0.02**	**99.86**	**21.64**	**31.86**
BART	0.42	97.78	2.30	82.00
+P	0.22	98.78	1.82	87.98
+NUP	0.64	96.52	1.34	91.40
+NUP+P	**0.20**	**99.02**	**0.76**	**95.32**

Table 4: Random permutation stress test of AT models.

	Model	1	2	3	Avg.
E2E	AT+NUP+P	61.5	16.5	22.0	**1.61**
	AT+P	30.0	44.0	26.0	1.96
	AT	25.0	49.5	25.5	2.01
ViGGO	AT+NUP+P	57.5	27.5	15.0	**1.58**
	AT+P	10.0	29.5	60.5	2.51
	AT	43.0	46.0	11.0	1.68

Table 5: Human Evaluation results. Table shows the percent of times each model was ranked 1 (best), 2, 3 (worst) in terms of naturalness and average rank.

the biGRU AT+NUP model, but otherwise helping performance.

Random Permutation Stress Test Results of the random permutation experiment are shown in Table 4. Overall, all models have an easier time following the neural utterance planner's reordering of the random permutations. Phrase training also generally improved semantic error rate. All models perform quite well on the E2E permutations. With phrase-training, all E2E models achieve less than 0.6% semantic error rate following random utterance plans. Starker differences emerge on the ViGGO dataset. The biGRU+NUP+P model achieves a 8.98% semantic error rate and only correctly follows the given order 64.5% of the time, which is a large decrease in performance compared to the ViGGO test set.

Human Evaluation Results of the human evaluation are shown in Table 5. We show the number of times each system was ranked 1 (most natural), 2, or 3 (least natural) and the average rank overall. Overall, we see that BART with the neural utterance planner and phrase-augmentation training is preferred on both datasets, suggesting that the utterance planner is producing natural orderings of the attribute-values, and the model can generate reasonable output for it. On the E2E dataset, we also see small differences in between the AT+P and AT models suggesting that when following an arbitrary ordering, the phrase-augmented model is about as natural as the non-phrase trained model. This is encouraging as the phrase trained model has lower semantic error rates. On the ViGGO dataset we do find that the phrase trained model is less natural, suggesting that in the small data setting, phrase-training may hurt fluency when trying to follow a difficult utterance plan.

For agreement we compute average Kendall's τ between each pair of annotators for each dataset. On E2E, we have $\tau = .853$ and ViGGO we have $\tau = .932$ suggesting very strong agreement.

7 Discussion

One consistently worrying sign throughout the first two experiments is that the automatic metrics are not good indicators of semantic correctness. For example the ROUGE-L score of the E2E AT ORACLE models is about 8 points higher than the AT+NUP models, but the AT+NUP models make fewer semantic errors. Other similar examples can be found where the automatic metric would suggest picking the more error prone model over another. As generating fluent text becomes less of a difficult a problem, these shallow ngram overlap methods will cease to suffice as distinguishing criteria.

The second experiments also reveal limitations in the controllable model's ability to follow arbitrary orderings. The biGRU and Transformer models in the small-data ViGGO setting are not able to generalize effectively on non-training distribution utterance plans. BART performance is much better here, but is still hovering around 2% semantic error rate and only roughly 88% of outputs conform to the intended utterance plan. Thankfully, if an exact ordering is not required, using the neural utterance planner to propose an order leads to more semantically correct outputs.

8 Limitations

While we are able to acheive very low test-set SER for both corpora, we should caution that this required extensive manual development of matching rules to produce MR/utterance alignments, which in turn resulted in significant cleaning of the training datasets. We chose to do this over pursuing a model based strategy of aligning utterance subspans to attribute-values because we wanted to better understand how systematically S2S models can represent arbitray order permutations independent of alignment model error.

Also we should note that data cleaning can yield more substantial decreases in semantic errors (Dušek et al., 2019; Wang, 2019) and is an important consideration in any practical neural NLG.

9 Related Work

MR linearizations for S2S models have been studied in a variety of prior works. Nayak et al. (2017) explore several ways of incorporating sentence planning into an MR linearization for S2S models, comparing a flat alignment order (equivalent to the alignment order used in this paper) against various sentence level groupings. Reed et al. (2018) add additional sentence and discourse structuring variables to indicate contrasts or sentential groupings. Balakrishnan et al. (2019) experiment both with tree structured MRs and encoders and compare them to linearized trees with standard S2S models. They also find that properly aligned linearization can lead to a controllable generator. These papers do not, however, explore how other linearization strategies compare in terms of faithfulness, and they do not evaluate the degree to which a S2S model can follow realization orders not drawn from the training distribution.

Castro Ferreira et al. (2017) compare a S2S NLG model using various linearizations of abstract meaning representation (AMR) graphs, including a model-based alignment very similar to the AT linearization presented in this work. However, they evaluate only on automatic quality measures and do not explicitly measure the semantic correctness of the generated text or the degree to which the model realizes the text in the order implied by the linearized input.

Works like Moryossef et al. (2019a,b) and Castro Ferreira et al. (2019) show that treating various planning tasks as separate components in a pipeline, where the components themselves are implemented with neural models, improves the overall quality and semantic correctness of generated utterances relative to a completely end-to-end neural NLG model. However, they do not test the systematicty of the neural generation components, i.e. the ability to perform correctly when given an arbitrary or random input from the preceding component, as we do here with the random permutation stress test.

Other papers mention linearization order anecdottally but do quantify its impact. For example, Juraska et al. (2018) experiment with random linearization orderings during development, but do not use them in the final model or report results using them, and Gehrmann et al. (2018) report that using a consistent linearization strategy worked best for their models but do not specify the exact order. Juraska et al. (2018) also used sentence level data augmentation, i.e. splitting a multi-sentence example in multiple single sentence examples, similar in spirit to our proposed phrase based method, but they do not evaluate its effect independently.

10 Conclusion

We present an empirical study on the effects of linearization order and phrase based data augmentation on controllable MR-to-text generation. Our findings support the importance of aligned linearization and phrase training for improving model control. Additionally, we identify limitations to this ability, specifically in the small data, random permutation setting, and will focus on this going forward.

Acknowledgments

We would like to thank the anonymous reviewers, as well as Katy Gero and Faisal Ladhak for their comments and discussion when drafting this paper. We would also like to thank Juraj Juraska for making the outputs of their ViGGO models available.

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

Mira Ariel. 2001. Accessibility theory: An overview. In *Text Representation: Linguistic and psycholinguistic aspects*, volume 8, pages 29–87. John Benjamins.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019. Constrained Decoding for Neural NLG from Compositional Representations in Task-Oriented Dialogue. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 831–844, Florence, Italy. Association for Computational Linguistics.

Thiago Castro Ferreira, Iacer Calixto, Sander Wubben, and Emiel Krahmer. 2017. Linguistic realisation as machine translation: Comparing different MT models for AMR-to-text generation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 1–10, Santiago de Compostela, Spain. Association for Computational Linguistics.

Thiago Castro Ferreira, Chris van der Lee, Emiel van Miltenburg, and Emiel Krahmer. 2019. Neural data-to-text generation: A comparison between pipeline and end-to-end architectures. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 552–562, Hong Kong, China. Association for Computational Linguistics.

Stanley F. Chen and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *34th Annual Meeting of the Association for Computational Linguistics*, pages 310–318, Santa Cruz, California, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, 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, Doha, Qatar. Association for Computational Linguistics.

Ondřej Dušek, David M. Howcroft, and Verena Rieser. 2019. Semantic Noise Matters for Neural Natural Language Generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 421–426, Tokyo, Japan. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the state-of-the-art of End-to-End Natural Language Generation: The E2E NLG challenge. *Computer Speech & Language*, 59:123 – 156.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander Rush. 2018. End-to-End Content and Plan Selection for Data-to-Text Generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56, Tilburg University, The Netherlands. Association for Computational Linguistics.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A Framework for Modeling the Local Coherence of Discourse. *Computational Linguistics*, 21(2):203–225.

Juraj Juraska, Kevin Bowden, and Marilyn Walker. 2019. ViGGO: A Video Game Corpus for Data-To-Text Generation in Open-Domain Conversation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 164–172, Tokyo, Japan. Association for Computational Linguistics.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A Deep Ensemble Model with Slot Alignment for Sequence-to-Sequence Natural Language Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 152–162, New Orleans, Louisiana. Association for Computational Linguistics.

M. G. Kendall. 1938. A NEW MEASURE OF RANK CORRELATION. *Biometrika*, 30(1-2):81–93.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pretraining for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-Based Statistical Language Generation Using Graphical Models and Active Learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1552–1561, Uppsala, Sweden. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019a. Improving Quality and Efficiency in Plan-based Neural Data-to-text Generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 377–382, Tokyo, Japan. Association for Computational Linguistics.

Amit Moryossef, Yoav Goldberg, and Ido Dagan. 2019b. Step-by-Step: Separating Planning from Realization in Neural Data-to-Text Generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota. Association for Computational Linguistics.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

Neha Nayak, Dilek Hakkani-Tür, Marilyn Walker, and Larry Heck. 2017. To Plan or not to Plan? Discourse Planning in Slot-Value Informed Sequence to Sequence Models for Language Generation. In *Proceedings of Interspeech 2017*, pages 3339–3343.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E Dataset: New Challenges For End-to-End Generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Lena Reed, Shereen Oraby, and Marilyn Walker. 2018. Can Neural Generators for Dialogue Learn Sentence Planning and Discourse Structuring? In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 284–295, Tilburg University, The Netherlands. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Studies in Natural Language Processing. Cambridge University Press.

Alexander Rush. 2018. The Annotated Transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia. Association for Computational Linguistics.

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

Matthew Stone, Christine Doran, Bonnie Webber, Ton ia Bleam, and Martha Palmer. 2003. Microplanning with Communicative Intentions: The SPUD System. *Computational Intelligence*, 19(4):311–381.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Marilyn A. Walker, Owen Rambow, and Monica Rogati. 2001. SPoT: A Trainable Sentence Planner. In *Second Meeting of the North American Chapter of the Association for Computational Linguistics*.

Hongmin Wang. 2019. Revisiting Challenges in Data-to-Text Generation with Fact Grounding. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 311–322, Tokyo, Japan. Association for Computational Linguistics.

Liang Wang, Wei Zhao, Ruoyu Jia, Sujian Li, and Jingming Liu. 2019. Denoising based Sequence-to-Sequence Pre-training for Text Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4003–4015, Hong Kong, China. Association for Computational Linguistics.

B. L. Welch. 1947. THE GENERALIZATION OF 'STUDENT'S' PROBLEM WHEN SEVERAL DIFFERENT POPULATION VARLANCES ARE INVOLVED. *Biometrika*, 34(1-2):28–35.

Tsung-Hsien Wen, Milica Gašić, 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, Lisbon, Portugal. Association for Computational Linguistics.

A Models and Hyper-paramter Search Details

A.1 General Details

Utterance text was sentence and word tokenized, and all tokens were lower-cased. A special *sentence-boundary* token was inserted between sentences. All words occurring fewer than 3 times on the training set were replaced with a special *unknown* token. We used a batch size of 128 for all biGRU and Transformer models. All models were trained on a single Nvidia Tesla v100 for at most 700 epochs.

Delexicalization The ViGGO corpus is relatively small and the attributes *name*, *developer*, *release_year*, *expected_release_date*, and *specifier* can have values that are only seen several times during training. Neural models often struggle to learn good representations for infrequent inputs, which can, in turn, lead to poor test-set generalization. To alleviate this, we delexicalize these values in the utterance. That is, we replace them with an attribute specific placeholder token.

Additionally, for *specifier* whose values come from the open class of adjectives, we represent the specified adjective with a placeholder which marks two features, whether it is consonant (C) or vowel initial (V) (e.g. "dull" vs. "old") and whether it is in regular (R) or superlative (S) form (e.g. "dull" vs. "dullest") since these features can effect the surrounding context in which the adjective is realized. See the following lexicalized/delexicalized examples:

- *specifier* = "oldest" – vowel initial, superlative
  - *What is the oldest game you've played?*
  - *What is the SPECIFIER_V_S game you've played?*
- *specifier* = "old" – vowel initial, regular
  - *What is an old game you've played?*
  - *What is an SPECIFIER_V_R game you've played?*
- *specifier* = "new" – consonant initial, regular
  - *What is a new game you've played?*
  - *What is a SPECIFIER_C_R game you've played?*

All generated delexicalized utterances are post-processed with the corresponding attribute-values before computing evaluation metrics (i.e., they are re-lexicalized with the appropriate value strings from the input MR).

A.2 biGRU Model Definition

Let $\mathcal{V}$ be the encoder input vocabulary, and $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times D_w}$ an associated word embedding matrix where $\mathbf{E}_x \in \mathbb{R}^{D_w}$ denotes the D_w-dimensional embedding for each $x \in \mathcal{V}$. Given a linearized MR $\pi(\mu) = \mathbf{x} = \left[a, x_1, x_2, \ldots, x_{|\mu|}\right] \in \mathcal{V}^m$ where the length of the sequence is $m = |\mu| + 1$, let $\mathbf{v}_i = \mathbf{E}_{\mathbf{x}_i}$ for $i \in \{1, \ldots m\}$.

The hidden states of the first GRU encoder layer are computed as

$$\begin{aligned}
\hat{\boldsymbol{h}}_0^{(1)} &= \check{\boldsymbol{h}}_{m+1}^{(1)} = \mathbf{0} \\
\hat{\boldsymbol{h}}_i^{(1)} &= \operatorname{GRU}\left(\mathbf{v}_i, \hat{\boldsymbol{h}}_{i-1}^{(1)}; \hat{\eta}^{(1)}\right) \quad \text{for } i \in 1, \ldots, m \\
\check{\boldsymbol{h}}_i^{(1)} &= \operatorname{GRU}\left(\mathbf{v}_i, \check{\boldsymbol{h}}_{i+1}^{(1)}; \check{\eta}^{(1)}\right) \quad \text{for } i \in m, \ldots, 1 \\
\boldsymbol{h}_i^{(1)} &= \left[\hat{\boldsymbol{h}}_i, \check{\boldsymbol{h}}_i\right]
\end{aligned}$$

where $[\cdot]$ is the concatenation operator, $\hat{\boldsymbol{h}}_i^{(1)}, \check{\boldsymbol{h}}_i^{(1)} \in \mathbb{R}^{D_h}$, $\boldsymbol{h}_i^{(1)} \in \mathbb{R}^{2D_h}$, and $\hat{\eta}^{(1)}$ and $\check{\eta}^{(1)}$ are the forward and backward encoder GRU parameters.

When using a two layer GRU, we similarly compute

$$\begin{aligned}
\hat{\boldsymbol{h}}_0^{(2)} &= \check{\boldsymbol{h}}_{m+1}^{(2)} = \mathbf{0} \\
\hat{\boldsymbol{h}}_i^{(2)} &= \operatorname{GRU}\left(\boldsymbol{h}_i^{(1)}, \hat{\boldsymbol{h}}_{i-1}^{(2)}; \hat{\eta}^{(2)}\right) \quad \text{for } i \in 1, \ldots, m \\
\check{\boldsymbol{h}}_i^{(2)} &= \operatorname{GRU}\left(\boldsymbol{h}_i^{(1)}, \check{\boldsymbol{h}}_{i+1}^{(2)}; \check{\eta}^{(2)}\right) \quad \text{for } i \in m, \ldots, 1 \\
\boldsymbol{h}_i^{(2)} &= \left[\hat{\boldsymbol{h}}_i, \check{\boldsymbol{h}}_i\right]
\end{aligned}$$

where $\hat{\boldsymbol{h}}_i^{(2)}, \check{\boldsymbol{h}}_i^{(2)} \in \mathbb{R}^{D_h}$, $\boldsymbol{h}_i^{(2)} \in \mathbb{R}^{2D_h}$, and $\hat{\eta}^{(2)}$ and $\check{\eta}^{(2)}$ are the forward and backward encoder GRU parameters for the second layer.

Going forward, let $\boldsymbol{h}_i$ correspond to the final encoder output, i.e. $\boldsymbol{h}_i = \boldsymbol{h}_i^{(1)}$ in the one-layer biGRU case, and $\boldsymbol{h}_i = \boldsymbol{h}_i^{(2)}$ in the two layer case.

Let $\mathcal{W}$ be the vocabulary of utterance tokens, and $\mathbf{D} \in \mathbb{R}^{|\mathcal{W}| \times D_w}$ an associated embedding matrix, where $\mathbf{D}_y \in \mathbb{R}^{D_w}$ denotes a D_w-dimensional embedding for each $y \in \mathcal{W}$.

Given the decoder input sequence $\mathbf{y} = y_1, y_2, \ldots, y_{|\mathbf{y}|}$, let $\mathbf{w}_i = \mathbf{D}_{y_i}$ for $i \in \{1, \ldots n\}$. where $n = |\mathbf{y}| - 1$

We compute the hidden states of the i-th layer

	Model	Layers	LS	WD	Optim.	LR	Emb.	Attention	Params	Train Time
E2E	RND	2	0.1	10^{-5}	Adam	10^{-5}	untied	Bahdanau	14,820,419	31.16
	FP	2	0.1	10^{-5}	SGD	0.1	untied	Bahdanau	14,820,003	24.78
	IF	2	0.1	0.0	SGD	0.5	untied	General	14,557,763	26.44
	IF+P	2	0.1	0.0	SGD	0.5	untied	General	14,557,763	15.23
	AT	2	0.1	10^{-5}	Adam	10^{-5}	untied	Bahdanau	14,820,419	26.07
	AT+P	2	0.1	10^{-5}	Adam	10^{-5}	untied	Bahdanau	14,820,419	36.49
ViGGO	RND	2	0.1	10^{-5}	SGD	0.25	untied	General	14,274,865	20.69
	FP	1	0.1	10^{-5}	Adam	10^{-5}	untied	Bahdanau	7,718,193	30.07
	IF	1	0.0	0.0	SGD	0.5	untied	Bahdanau	7,712,049	11.62
	IF+	1	0.0	0.0	SGD	0.5	untied	Bahdanau	7,712,049	5.08
	AT	2	0.1	0.0	Adam	10^{-5}	untied	Bahdanau	14,537,521	21.01
	AT+P	2	0.1	0.0	Adam	10^{-5}	untied	Bahdanau	14,537,521	14.95

Table 6: Winning hyperparameter settings for biGRU models. LS and WD indicate label smoothing and weight decay respectively. Train time is in hours.

of the decoder as,

$$\boldsymbol{g}_0^{(i)} = \tanh\left(\mathbf{W}^{(i)}\boldsymbol{h}_m^{(i)} + \mathbf{b}^{(i)}\right)$$

for $j \in 1, \ldots, n$

$$\boldsymbol{g}_j^{(i)} = \mathrm{GRU}\left(\boldsymbol{g}_j^{(i-1)}, \boldsymbol{g}_{j-1}^{(i)}; \zeta^{(i)}\right)$$

where $\boldsymbol{g}_j^{(0)} = \mathbf{w}_j$, $\boldsymbol{g}_j^{(i)} \in \mathbb{R}^{D_h}$, $\mathbf{W}^{(i)} \in \mathbb{R}^{D_h \times 2D_h}$, $\mathbf{b}^{(i)} \in \mathbb{R}^{D_h}$ and $\zeta^{(i)}$ are the decoder GRU parameters.

Going forward, let $\boldsymbol{g}_i$ correspond to the final decoder output, i.e. $\boldsymbol{g}_i = \boldsymbol{g}_i^{(1)}$ in the one-layer biGRU case, and $\boldsymbol{g}_i = \boldsymbol{g}_i^{(2)}$ in the two layer case.

Then the decoder states attend to the encoder states,

$$\bar{\boldsymbol{h}}_i = \sum_{j=1}^{m} \alpha_{i,j}\boldsymbol{h}_j \qquad \text{for } i \in 1, \ldots, n$$

where $\alpha_{i,j} \in (0, 1)$ is the attention weight of decoder state i on encoder state j and $\sum_{j=1}^{m} \alpha_{i,j} = 1$. We compute attention in one of two ways (the attention method is a hyperparemeter option):

1. Feed-forward "Bahdanau" style attention (Bahdanau et al., 2015), also known as "concat" (Luong et al., 2015):

   $$\alpha_{i,j} = \mathbf{k}\tanh\left(\mathbf{K}\begin{bmatrix}\boldsymbol{g}_i \\ \boldsymbol{h}_j\end{bmatrix}\right)$$

   with $\mathbf{K} \in \mathbb{R}^{D_h \times 3D_h}$ and $\mathbf{k} \in \mathbb{R}^{D_h}$.

2. "general" (Luong et al., 2015) :

   $$\alpha_{i,j} = \boldsymbol{g}_i\mathbf{K}\boldsymbol{h}_j$$

   with $\mathbf{K} \in \mathbb{R}^{D_h \times 2D_h}$.

Finally, for $i \in 1, \ldots, n$ we compute

$$\mathbf{z}_i = \tanh\left(\mathbf{W}^{(z)}\begin{bmatrix}\boldsymbol{g}_i \\ \bar{\boldsymbol{h}}_i\end{bmatrix} + \mathbf{b}^{(z)}\right)$$

and

$$p\left(y_{i+1}|y_{\leq i}, \pi(\mu)\right) =$$
$$\mathrm{softmax}\left(\mathbf{W}^{(o)}\mathbf{z}_i + \mathbf{b}^{(o)}\right)_{y_{i+1}}$$

where $\mathbf{W}^{(z)} \in \mathbb{R}^{D_h \times 3D_h}$, $\mathbf{b}^{(z)} \in \mathbb{R}^{D_h}$, $\mathbf{b}^{(o)} \in \mathbb{R}^{|\mathcal{W}|}$, and $\mathbf{W}^{(o)} \in \mathbb{R}^{|\mathcal{W}| \times D_h}$ is the output embedding matrix. As a hyperparamter setting, we consider tieing the decoder input and output embedding matrices, i.e. $\mathbf{D} = \mathbf{W}^{(o)}$. Dropout of 0.1 is applied to all embedding, GRU outputs, and linear layer outputs. We set $D_w = D_h = 512$.

A.3 biGRU Hyperparameter Search

We grid-search over the following hyperparameter values:

- **Layers:** 1, 2
- **Label Smoothing:** 0, 0.1
- **Weight Decay:** 0, 10^{-5}

- **Optimizer/Learning Rate:** Adam/10^{-3}, Adam/10^{-4}, Adam/10^{-5}, SGD/0.5, SGD/0.25, SGD/0.1
- **Tie Decoder Embeddings:** tied, untied
- **Attention:** Bahdanau, General

During hyperparameter search, we train for at most 500 epochs, evaluating BLEU every 25 epochs to select the best model. We decay the learning if validation log-likelihood stops increasing for five epochs. We decay the learning rate by $lr^{i+1} = .99 \times lr^i$.

Winning hyperparameter settings are presented Table 6.

A.4 Transformer Model Definition

Each Transformer layer is divided into blocks which each have three parts, (i) layer norm, (ii) feed-forward/attention, and (iii) skip-connection. We first define the components used in the transformer blocks before describing the overall S2S transformer. Starting with layer norm (Ba et al., 2016), let $\mathbf{H} \in \mathbb{R}^{m\times n}$, then we have $\mathrm{LN} : \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$,

$$\mathrm{LN}(\mathbf{H}; \mathbf{a}, \mathbf{b}) = \mathbf{A} \odot (\mathbf{H} - \boldsymbol{\mu}) \odot \Lambda + \mathbf{b}$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ are learned parameters, $\odot$ is the elementwise product, $\mathbf{A} = [\mathbf{a}, \ldots, \mathbf{a}] \in \mathbb{R}^{m\times n}$ is a tiling of the parameter vector, $\mathbf{a}$, m times, and $\boldsymbol{\mu}, \boldsymbol{\Lambda} \in \mathbb{R}^{m\times n}$ are defined elementwise as

$$\boldsymbol{\mu}_{i,j} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{H}_{i,k}$$

and

$$\boldsymbol{\Lambda}_{i,j} = \left(\sqrt{\frac{1}{n-1}\sum_{k=1}^{n}\left(\mathbf{H}_{i,k} - \boldsymbol{\mu}_{i,j}\right)^2 + \epsilon}\right)^{-1}$$

respectively. The ϵ term is a small constant for numerical stability, set to 10^{-5}.

The inplace feed-forward layer, FF, is a simple single-layer perceptron with ReLU activation ($\mathrm{ReLU}(\mathbf{H}) = \max(\mathbf{0}, \mathbf{H})$) (Nair and Hinton, 2010), applied to each row of an $m \times n$ input matrix, i.e. a sequence of m objects with n features,

$$\mathrm{FF}\left(\mathbf{H}; \mathbf{W}^{(i)}, \mathbf{W}^{(j)}, \mathbf{b}^{(i)}, \mathbf{b}^{(j)}\right) =$$
$$\mathrm{ReLU}\left(\mathbf{H}\mathbf{W}^{(i)} + \mathbf{b}^{(i)}\right)\mathbf{W}^{(j)} + \mathbf{b}^{(j)}$$

where $\mathbf{W}^{(i)} \in \mathbb{R}^{D_w\times D_h}$, $\mathbf{b}^{(i)} \in \mathbb{R}^{D_h}$, $\mathbf{W}^{(j)} \in \mathbb{R}^{D_h\times D_w}$, $\mathbf{b}^{(j)} \in \mathbb{R}^{D_w}$ are learned parameters and matrix-vector additions (i.e. $\mathbf{X} + \mathbf{b}$) are broadcast across the matrix rows.

The final component to be defined is the multi-head attention, MultiAttn which is defined as

$$\mathrm{MultiAttn}(\mathbf{Q}, \mathbf{K}; \mathbf{W}^{(a_1)}, \mathbf{W}^{(a_2)}) =$$

$$\begin{bmatrix} \mathrm{Attn}\left(\mathbf{Q}\mathbf{W}^{(a_1)}_{1,1}, \mathbf{K}\mathbf{W}^{(a_1)}_{2,1}, \mathbf{K}\mathbf{W}^{(a_1)}_{3,1}\right), \\ \vdots \\ \mathrm{Attn}\left(\mathbf{Q}\mathbf{W}^{(a_1)}_{1,H}, \mathbf{K}\mathbf{W}^{(a_1)}_{2,H}, \mathbf{K}\mathbf{W}^{(a_1)}_{3,H}\right) \end{bmatrix} \mathbf{W}^{(a_2)}$$

where $[\cdot]$ indicates column-wise concatenation, $\mathbf{W}^{(a_1)}_{1,*} \in \mathbb{R}^{D_w\times D_w/H}$ and $\mathbf{W}^{(a_2)} \in \mathbb{R}^{D_w\times D_w}$ are learned parameters, H is the number of attention heads, and Attn is defined,

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{D_w}}\right)\mathbf{V}.$$

Additionally, there is a masked variant of attention, $\mathrm{MultiAttn}_M$ where the attention is computed

$$\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T \odot \mathbf{M}}{\sqrt{D_w}}\right)\mathbf{V}$$

where $\mathbf{M} \in \mathbb{R}^{n\times m}$ is a lower triangular matrix, i.e. values on or below the diagonal are 1 and all other values are $-\infty$.

Given these definitions, we now define the S2S transformer. Let $\mathcal{V}$ be the encoder input vocabulary, and $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}|\times D_w}$ an associated word embedding matrix where $\mathbf{E}_x \in \mathbb{R}^{D_w}$ denotes the D_w-dimensional embedding for each $x \in \mathcal{V}$. Given a linearized MR $\pi(\mu) = \mathbf{x} = \left[a, x_1, x_2, \ldots, x_{|\mu|}\right] \in \mathcal{V}^m$ where the length of the sequence is $m = |\mu| + 1$, let $\mathbf{v}_i = \mathbf{E}_{\mathbf{x}_i}$ for $i \in \{1, \ldots m\}$.

Additionally let $\mathbf{P} \in \mathbb{R}^{m_{max}\times D_w}$ be a sinusoidal position embedding matrix defined elementwise with

$$\mathbf{P}_{i,2j} = \sin\left(\frac{i}{10,000^{\frac{2j}{D_w}}}\right)$$
$$\mathbf{P}_{i,2j+1} = \cos\left(\frac{i}{10,000^{\frac{2j}{D_w}}}\right).$$

The encoder input sequence $\mathbf{H}^{(0)} \in \mathbb{R}^{m \times D_w}$ is then defined by

$$\mathbf{H}^{(0)} = \begin{bmatrix} \mathbf{v}_1 + \mathbf{P}_1, \\ \mathbf{v}_2 + \mathbf{P}_2, \\ \vdots \\ \mathbf{v}_m + \mathbf{P}_m \end{bmatrix}$$

A sequence of l transformer encoder layers are then applied to the encoder input, i.e. $\mathbf{H}^{(i+1)} = \mathrm{TF}^{(i)}_{enc}\left(\mathbf{H}^{(i)}\right)$. Each encoder transformer layer computes the following,

(Self-Attention Block)

$$\check{\mathbf{H}}^{(i)} = \mathrm{LN}\left(\mathbf{H}^{(i)}; \mathbf{a}^{(i,1)}, \mathbf{b}^{(i,1)}\right)$$

$$\bar{\mathbf{H}}^{(i)} = \mathrm{MultiAttn}\left(\check{\mathbf{H}}^{(i)}, \check{\mathbf{H}}^{(i)}; \mathbf{W}^{(i,a_1)}, \mathbf{W}^{(i,a_2)}\right)$$

$$\hat{\mathbf{H}}^{(i)} = \mathbf{H}^{(i)} + \bar{\mathbf{H}}^{(i)}$$

(Feed-Forward Block)

$$\dot{\mathbf{H}}^{(i)} = \mathrm{LN}\left(\hat{\mathbf{H}}^{(i)}; \mathbf{a}^{(i,2)}, \mathbf{b}^{(i,2)}\right)$$

$$\ddot{\mathbf{H}}^{(i)} = \mathrm{FF}\left(\dot{\mathbf{H}}^{(i)}; \mathbf{W}^{(i,1)}, \mathbf{W}^{(i,2)}, \mathbf{b}^{(i,1)}, \mathbf{b}^{(i,2)}\right)$$

$$\mathbf{H}^{(i+1)} = \hat{\mathbf{H}}^{(i)} + \ddot{\mathbf{H}}^{(i)}$$

We denote the final encoder output for l layers as $\mathbf{H} = \mathbf{H}^{(l)}$.

Let $\mathcal{W}$ be the vocabulary of utterance tokens, and $\mathbf{D} \in \mathbb{R}^{|\mathcal{W}| \times D_w}$ an associated embedding matrix, where $\mathbf{D}_y \in \mathbb{R}^{D_w}$ denotes a D_w-dimensional embedding for each $y \in \mathcal{W}$.

Given the decoder input sequence $\mathbf{y} = y_1, y_2, \ldots, y_{|\mathbf{y}|}$, let $\mathbf{w}_i = \mathbf{D}_{y_i}$ for $i \in \{1, \ldots n\}$. where $n = |\mathbf{y}| - 1$

$$\mathbf{G}^{(0)} = \begin{bmatrix} \mathbf{w}_1 + \mathbf{P}_1, \\ \mathbf{w}_2 + \mathbf{P}_2, \\ \vdots \\ \mathbf{w}_n + \mathbf{P}_n \end{bmatrix}.$$

A sequence of l transformer decoder layers are then applied to the decoder input, i.e. $\mathbf{G}^{(i+1)} = \mathrm{TF}^{(i)}_{dec}\left(\mathbf{G}^{(i)}\right)$. Each decoder transformer layer computes the following,

(Masked Self-Attention Block)

$$\check{\mathbf{G}}^{(i)} = \mathrm{LN}\left(\mathbf{G}^{(i)}; \mathbf{a}^{(i,1)}, \mathbf{b}^{(i,1)}\right)$$

$$\bar{\mathbf{G}}^{(i)} = \mathrm{MultiAttn}_M\left(\check{\mathbf{G}}^{(i)}, \check{\mathbf{G}}^{(i)}; \mathbf{W}^{(i,a_1)}, \mathbf{W}^{(i,a_2)}\right)$$

$$\hat{\mathbf{G}}^{(i)} = \mathbf{G}^{(i)} + \bar{\mathbf{G}}^{(i)}$$

(Encoder-Attention Block)

$$\grave{\mathbf{G}}^{(i)} = \mathrm{LN}\left(\hat{\mathbf{G}}^{(i)}; \mathbf{a}^{(i,2)}, \mathbf{b}^{(i,2)}\right)$$

$$\tilde{\mathbf{G}}^{(i)} = \mathrm{MultiAttn}\left(\grave{\mathbf{G}}^{(i)}, \mathbf{H}; \mathbf{W}^{(i,a_3)}, \mathbf{W}^{(i,a_4)}\right)$$

$$\acute{\mathbf{G}}^{(i)} = \hat{\mathbf{G}}^{(i)} + \tilde{\mathbf{G}}^{(i)}$$

(Feed-Forward Block)

$$\dot{\mathbf{G}}^{(i)} = \mathrm{LN}\left(\acute{\mathbf{G}}^{(i)}; \mathbf{a}^{(i,3)}, \mathbf{b}^{(i,3)}\right)$$

$$\ddot{\mathbf{G}}^{(i)} = \mathrm{FF}\left(\dot{\mathbf{G}}^{(i)}; \mathbf{W}^{(i,1)}, \mathbf{W}^{(i,2)}, \mathbf{b}^{(i,1)}, \mathbf{b}^{(i,2)}\right)$$

$$\mathbf{G}^{(i+1)} = \acute{\mathbf{G}}^{(i)} + \ddot{\mathbf{G}}^{(i)}$$

Let the $\mathbf{G} = \mathbf{G}^{(l)}$ denote the final decoder output, and let $\mathbf{g}_i$ be the i-th row of $\mathbf{G}$ corresponding to the decoder representation of the i-th decoder state. The probability of the next word is

$$p\left(y_{i+1} | y_{\leq i}, \pi(\mu)\right) = \mathrm{softmax}\left(\mathbf{W}^{(o)}\mathbf{g}_i + \mathbf{b}^{(o)}\right)_{y_{i+1}}$$

where $\mathbf{W}^{(o)} \in \mathbb{R}^{|\mathcal{W}| \times D_w}$ and $\mathbf{b}^{(o)} \in \mathbb{R}^{D_w}$ are learned parameters.

The input embedding dimension is $D_w = 512$ and inner hidden layer size is $D_h = 2048$. The encoder and decoder have separate parameters. We used $H = 8$ heads in all multi-head attention layers. We used Adam with the learning rate schedule provided in Rush (2018) (factor=1, warmup=8000). Dropout was set to 0.1 was applied to input embeddings and each skip connection (i.e. the third line in each block definition). As a hyperparameter, we optionally tie the decoder input and output embeddings, i.e. $\mathbf{D} = \mathbf{W}^{(o)}$.

	Model	Layers	LS	Emb.	Params	Train Time
E2E	RND	1	0.1	tied	7,966,787	18.09
	FP	1	0.1	tied	7,970,371	17.30
	IF	1	0.1	untied	8,525,379	17.52
	IF+P	1	0.1	untied	8,525,379	28.11
	AT	2	0.1	untied	15,881,795	23.73
	AT+P	2	0.1	untied	15,881,795	29.39
ViGGO	RND	2	0.0	untied	15,598,897	11.22
	FP	2	0.1	untied	15,605,041	9.68
	IF	2	0.1	untied	15,598,897	11.35
	IF+P	2	0.1	untied	15,598,897	9.09
	AT	2	0.1	untied	15,598,897	7.26
	AT+P	2	0.1	untied	15,598,897	5.87

Table 7: Winning hyperparameter settings for Transformer models (trained from scratch). LS indicates label smoothing. Train time is in hours.

A.5 Transformer Hyperparameter Search

We grid searched over the following Transformer hyper-parameters:

- **Tied Decoder Embeddings:** tied, untied
- **Layers:** 1, 2
- **Label Smoothing:** 0.0, 0.1

During hyperparameter search, we train for at most 500 epochs, evaluating BLEU every 25 epochs to select the best model.

Winning hyperparameter settings are presented Table 7.

A.6 BART Model Hyperparameters

We use the same settings as the fine-tuning for the CNN-DailyMail summarization task, although we modify the maximum number of updates to be roughly to be equivalent to 10 epochs on the training set when using a 500 token batch size, since the number of updates effects the learning rate scheduler. We selected the model iterate with lowest validation set cross-entropy.

While BART is unlikely to have seen any linearized MR in its pretraining data, its use of subword encoding allows it to encode arbitrary strings. Rather than extending it's encoder input vocabulary to add the MR tokens, we simply format the input MR as a string (in the correpsonding linearization order), e.g. "inform rating=good name=NAME platforms=PC platforms=Xbox".

	Model	B↑	R↑	SER↓	OA↑
biGRU	Rnd	47.8	59.5	1.84	—
	Fp	49.1	60.3	1.90	—
	If	48.4	59.9	3.86	—
	If+P	48.1	60.0	1.12	—
	At+BgUP	44.2	57.4	0.14	99.0
	At+NUP	48.8	61.2	0.02	99.8
	At+NUP+P	48.7	61.1	0.02	99.8
	At Oracle	56.9	73.2	0.18	99.2
Transformer	Rnd	47.8	59.4	2.26	—
	Fp	49.2	60.6	2.84	—
	If	49.7	60.9	1.46	—
	If+P	48.9	60.7	1.00	—
	At+BgUP	44.9	57.4	0.28	98.3
	At+NUP	49.3	61.3	0.10	99.4
	At+NUP+P	48.8	61.1	0.08	99.6
	At Oracle	56.9	73.1	0.48	97.5
BART	Rnd	46.9	59.5	0.48	—
	Fp	48.6	60.3	0.04	—
	If	48.2	60.3	0.18	—
	If+P	47.8	60.1	0.46	—
	At+BgUP	45.9	57.3	0.00	99.9
	At+NUP	49.0	61.2	0.02	99.8
	At+NUP+P	48.8	61.1	0.02	99.8
	At Oracle	55.7	72.8	0.20	99.0

Table 8: E2E validation set (B) Bleu, (R) Rouge-L, SER, and OA.

	Model	B↑	R↑	SER↓	OA↑
biGRU	Rnd	50.2	62.0	14.26	—
	Fp	50.5	62.3	16.68	—
	If	51.5	62.7	19.28	—
	If+P	49.6	61.7	12.88	—
	At+BgUP	49.8	60.5	2.76	91.1
	At+NUP	52.9	64.4	1.52	93.8
	At+NUP+P	52.7	63.9	1.40	94.3
	At Oracle	55.7	67.4	1.82	92.7
Transformer	Rnd	52.6	63.4	8.96	—
	Fp	52.8	63.5	8.48	—
	If	53.3	63.5	7.00	—
	If+P	52.5	63.3	3.88	—
	At+BgUP	48.8	60.3	3.72	80.3
	At+NUP	51.5	63.5	2.84	88.1
	At+NUP+P	51.6	63.4	2.60	88.5
	At Oracle	53.4	66.2	3.78	82.7
BART	Rnd	45.2	57.4	2.08	—
	Fp	46.2	58.8	1.86	—
	If	44.9	57.0	2.12	—
	If+P	48.4	60.0	2.26	—
	At+BgUP	44.8	56.1	1.00	95.5
	At+NUP	47.5	60.2	0.90	96.1
	At+NUP+P	49.0	61.1	1.02	95.7
	At Oracle	48.7	62.8	1.44	94.1

Table 9: ViGGO validation set (B) Bleu, (R) Rouge-L, SER, and OA.

A.7 Validation Results

Validation set results are shown in Table 8 and Table 9 for the E2E and ViGGO datasets respectively. Unlike the test results, reported in the main paper and appendix, validation SER and OA are computed automatically and not manually validated. All results are the average of five random initializations. Also we use the corrected MR produced by our attribute-value matching rules as input, rather than the original validation set MR.

B Neural Utterance Planner Model and Hyper-Parameter Search

We use the same general recurrent neural network model as defined in §A.2 with Bahdanau style attention (Bahdanau et al., 2015) to implement the neural utterance planner model. We trained for at most 50 epochs with batch size 128. We used the Adam optimizer with 0.0 weight decay. Decoder input/output embeddings were not tied. Models used embeddings and hidden layers of 512 dimensions. Models were trained to map If inputs to At outputs. We grid-searched over the following hyper-parameters:

Dataset	Model	Valid	Test
ViGGO	BgUP	0.417	0.347
	NUP	0.739	0.651
E2E	BgUP	0.433	0.432
	NUP	0.502	0.447

Table 10: Validation and test set Kendall's τ for BgUP and NUP models.

- **Layers:** 1, 2
- **Learning Rate:** $10^{-3}, 10^{-4}, 10^{-5}$
- **RNN cell:** GRU, LSTM
- **Bidirectional Encoder:** uni, bi

- **Label Smoothing:** 0.0, 0.1

with the following winning settings determined by Kendall's τ on the validation set:

- E2E — 1 layers, biLSTM, lr $= 10^{-5}$, 0.1 label smoothing
- ViGGO — 1 layer, uniLSTM, lr $= 10^{-4}$, 0.1 label smoothing

Validation and test set Kendall's τ are shown in Table 10.

C Expanded Test Set Results

We show full automatic evaluation metrics from the E2E official evaluation script. E2E and ViGGO results are shown in Table 11 and Table 12 respectively. We also show full manual semantic evaluation results in Table 13 and Table 14 for E2E and ViGGO respectively. We break out the counts of missing, wrong, and added attributes used for SER calculation. Wrong attributes occur when an attribute is realized with the wrong value. Added attribute indicate the model realized an attribute-value that was not given in the input MR. Repeated attributes, even when specified in the input MR are included in added counts. We also include the percentage of utterances with correct semantics regardless of order (Perf.).

			Automatic Quality Metrics				
			BLEU	NIST	METEOR	ROUGE-L	CIDEr
biGRU	RND		66.82	8.696	44.46	68.26	2.248
	FP		63.40	8.414	42.32	65.64	2.032
	IF		59.24	7.996	38.74	62.66	1.608
		+P	65.82	8.604	45.10	68.14	2.238
	AT+BGUP		66.38	8.682	45.04	68.28	2.298
	AT+NUP		66.30	8.744	44.92	68.92	2.284
		+P	66.48	8.758	44.98	69.12	2.300
	AT ORACLE		**69.84**	**9.244**	**47.92**	**77.28**	**2.338**
Transformer	RND		67.36	8.722	44.86	68.20	2.296
	FP		67.44	8.722	44.26	68.70	2.246
	IF		67.12	8.706	44.96	68.10	2.284
		+P	66.80	8.674	45.04	68.30	2.306
	AT+BGUP		66.82	8.722	45.20	68.44	2.322
	AT+NUP		67.00	8.792	45.08	68.98	2.306
		+P	66.74	8.760	45.08	69.14	2.306
	AT ORACLE		**69.30**	**9.198**	**47.88**	**77.02**	**2.352**
BART	RND		66.46	8.652	45.54	68.34	2.316
	FP		65.54	8.594	45.18	67.18	2.312
	IF		65.62	8.608	45.26	67.38	**2.326**
		+P	65.92	8.660	45.24	68.18	2.316
	AT+BGUP		66.24	8.620	45.66	68.68	2.318
	AT+NUP		66.56	8.682	45.52	69.22	2.314
		+P	66.26	8.678	45.30	69.34	2.308
	AT ORACLE		**68.34**	**9.084**	**48.28**	**77.08**	2.282

Table 11: E2E test set automatic quality measures from the official E2E evaluation script.

			Automatic Quality Metrics				
			BLEU	NIST	METEOR	ROUGE-L	CIDEr
biGRU	RND		50.18	8.300	37.78	61.56	2.490
	FP		50.18	8.132	37.18	61.04	2.460
	IF		50.24	8.160	37.40	61.32	2.458
		+P	49.48	8.010	37.26	61.58	2.430
	AT+BGUP		48.52	7.946	37.32	58.48	2.466
	AT+NUP		51.84	8.252	38.48	62.56	2.618
		+P	52.40	8.084	38.34	62.66	2.594
	AT ORACLE		**54.08**	**8.504**	**39.38**	**65.48**	**2.698**
Transformer	RND		52.04	8.166	38.10	62.86	2.556
	FP		52.58	8.246	38.32	63.02	2.574
	IF		52.28	8.184	38.14	62.58	2.568
		+P	52.34	8.106	38.44	63.12	2.570
	AT+BGUP		48.70	8.174	37.50	59.22	2.438
	AT+NUP		51.60	8.352	38.52	62.42	2.592
		+P	51.06	8.138	38.12	62.00	2.512
	AT ORACLE		**53.18**	**8.508**	**39.12**	**64.96**	**2.662**
BART	RND		43.72	7.814	37.70	55.12	2.304
	FP		47.04	8.184	38.48	58.88	2.416
	IF		43.06	7.744	37.62	54.36	2.238
		+P	**49.06**	**8.284**	38.36	59.66	**2.454**
	AT+BGUP		43.76	7.888	37.38	53.98	2.338
	AT+NUP		45.46	8.034	37.84	57.62	2.368
		+P	48.50	8.248	38.04	59.24	2.454
	AT ORACLE		47.10	8.194	**38.50**	**60.40**	2.444

Table 12: ViGGO test set automatic quality measures from the official E2E evaluation script.

	Model		Manual Semantic Metrics						
			Missing	Wrong	Added	Total	SER↓	OA↑	Perf.↑
biGRU	RND		112.8	**0.0**	2.6	115.4	2.64	—	81.70
	FP		157.8	79.8	47.6	285.2	6.54	—	68.84
	IF		215.0	320.8	14.6	550.4	12.64	—	26.96
		+P	7.4	**0.0**	2.4	9.8	0.24	—	98.44
	AT+BGUP		11.4	**0.0**	**0.0**	11.4	0.26	98.18	98.18
	AT+NUP		10.8	**0.0**	**0.0**	10.8	0.26	98.30	98.30
		+P	**0.2**	**0.0**	**0.0**	**0.2**	**0.00**	**99.96**	**99.96**
	AT ORACLE		36.6	**0.0**	0.8	37.4	0.84	94.34	94.34
Transformer	RND		44.8	**0.0**	1.0	45.8	1.06	—	92.80
	FP		128.0	1.6	5.6	135.2	3.10	—	79.32
	IF		25.2	**0.0**	3.6	28.8	0.66	—	95.64
		+P	12.4	**0.0**	**0.0**	12.4	0.28	—	98.04
	AT+BGUP		0.2	**0.0**	**0.0**	0.2	**0.00**	99.94	99.96
	AT+NUP		**0.0**	**0.0**	**0.0**	**0.0**	**0.00**	**100.00**	**100.00**
		+P	0.2	**0.0**	**0.0**	0.2	**0.00**	99.96	99.96
	AT ORACLE		32.4	**0.0**	2.6	35.0	0.76	94.96	95.06
BART	RND		**0.0**	**0.0**	5.8	5.8	0.14	—	99.14
	FP		**0.0**	**0.0**	7.0	7.0	0.16	—	98.90
	IF		**0.0**	**0.0**	8.6	8.6	0.18	—	98.62
		+P	**0.0**	**0.0**	13.0	13.0	0.30	—	97.94
	AT+BGUP		**0.0**	2.2	6.6	8.8	0.20	98.60	98.60
	AT+NUP		**0.0**	2.0	6.6	8.6	0.20	98.64	98.64
		+P	**0.0**	**0.0**	**0.0**	**0.0**	**0.00**	**100.00**	**100.00**
	AT ORACLE		17.0	2.2	12.2	31.4	0.70	95.30	95.42

Table 13: E2E test set semantic errors.

	Model		Manual Semantic Metrics						
			Missing	Wrong	Added	Total	SER↓	OA↑	Perf.↑
biGRU	RND		85.8	9.8	89.6	185.2	12.56	—	70.10
	FP		121.8	27.2	103.6	252.6	17.12	—	60.56
	IF		124.0	9.4	149.4	282.8	19.20	—	62.14
		+P	93.2	5.0	85.0	183.2	12.46	—	70.36
	AT+BGUP		31.4	**4.8**	13.8	50.0	3.40	89.82	89.82
	AT+NUP		**7.2**	5.2	10.8	**23.2**	**1.58**	93.72	93.72
		+P	11.8	5.8	**6.2**	23.8	1.62	**94.32**	**94.32**
	AT ORACLE		12.2	12.6	11.0	35.8	2.42	92.22	92.34
Transformer	RND		90.6	11.4	40.2	142.2	9.62	—	70.98
	FP		88.0	16.6	23.8	128.4	8.70	—	72.24
	IF		76.8	10.6	23.2	110.6	7.50	—	74.88
		+P	48.4	5.2	**8.6**	62.2	4.24	—	85.62
	AT+BGUP		49.8	5.6	13.4	68.8	4.68	79.12	85.16
	AT+NUP		23.2	6.6	10.0	39.8	2.70	88.32	89.58
		+P	**21.8**	**3.4**	**8.6**	**33.8**	**2.28**	**89.80**	**91.60**
	AT ORACLE		39.6	10.8	9.8	60.2	4.08	83.02	85.92
BART	RND		0.8	3.2	17.6	21.6	1.50	—	94.52
	FP		1.0	3.0	21.0	25.0	1.68	—	93.26
	IF		**0.2**	2.2	25.2	27.6	1.86	—	92.82
		+P	2.6	2.8	20.8	26.2	1.78	—	93.60
	AT+BGUP		0.6	**1.2**	6.0	7.8	0.52	**98.30**	**98.36**
	AT+NUP		0.6	**1.2**	6.2	8.0	0.54	98.18	98.18
		+P	2.2	2.6	**2.0**	**6.8**	**0.46**	98.14	98.14
	AT ORACLE		1.6	1.8	8.6	12.0	0.82	97.16	97.34

Table 14: ViGGO test set semantic error evaluation.

D Random Permutation Examples

We show examples of random permutation utterance plans, and those plans reordered using the BGUP and NUP models. We then show model realizations of those plans.

Random Utterance Plan

1 *food* = "Japanese"
2 *family_friendly* = "yes"
3 *area* = "riverside"
4 *name* = "Blue Spice"
5 *customer_rating* = "high"

BGUP Utterance Plan

1 *name* = "Blue Spice"
2 *family_friendly* = "yes"
3 *food* = "Japanese"
4 *area* = "riverside"
5 *customer_rating* = "high"

NUP Utterance Plan

1 *name* = "Blue Spice"
2 *food* = "Japanese"
3 *area* = "riverside"
4 *family_friendly* = "yes"
5 *customer_rating* = "high"

biGRU AT (Random) There is a Japanese child friendly place in the riverside area called Blue Spice with a high customer rating.

biGRU AT (BGUP) Blue Spice is a kid friendly Japanese place in the riverside area with a high customer rating.

biGRU AT (NUP) Blue Spice serves Japanese food in the riverside area. It is kid friendly and has a high customer rating.

biGRU AT+P (Random) There is a Japanese child friendly place in the riverside area called Blue Spice which has a high customer rating.

biGRU AT+P (BGUP) Blue Spice is kid friendly and serves Japanese food. It is located in the riverside. Its customer rating is high.

biGRU AT+P (NUP) Blue Spice serves Japanese food in the riverside area. It is child friendly and has a high customer rating.

Transformer AT (Random) There is a Japanese kid friendly place in the riverside area called Blue Spice. It has a high customer rating.

Transformer AT (BGUP) Blue Spice is a kid friendly Japanese place in the riverside area with a high customer rating.

Transformer AT (NUP) Blue Spice serves Japanese food in the riverside area. It is kid friendly and has a high customer rating.

Transformer AT+P (Random) There is a Japanese child friendly establishment in the riverside area called Blue Spice. It has a high customer rating.

Transformer AT+P (BGUP) Blue Spice is a kid friendly Japanese place by the riverside with a high customer rating.

Transformer AT+P (NUP) Blue Spice serves Japanese food in the riverside area. It is child friendly and has a high customer rating.

BART AT (Random) There is a Japanese kid friendly place in the riverside area called Blue Spice that has a high customer rating.

BART AT (BGUP) Blue Spice is a kid friendly Japanese food place in the riverside area with a high customer rating.

BART AT (NUP) Blue Spice serves Japanese food in the riverside area. It is kid friendly and has a high customer rating.

BART AT+P (Random) There is a Japanese family friendly place in the riverside area called Blue Spice with a high customer rating.

BART AT+P (BGUP) Blue Spice is a kid friendly Japanese food place in riverside with a high customer rating.

BART AT+P (NUP) Blue Spice serves Japanese food in the riverside area. It is child friendly and has a high customer rating.

Random Utterance Plan
1 *genres* = “arcade”
2 *player_perspective* = “first person”
3 *has_mac_release* = “no”
4 *platforms* = “Nintendo”
5 *name* = “The Legend of Zelda: Ocarina of Time”

BGUP Utterance Plan
1 *player_perspective* = “first person”
2 *genres* = “arcade”
3 *platforms* = “Nintendo”
4 *name* = “The Legend of Zelda: Ocarina of Time”
5 *has_mac_release* = “no”

NUP Utterance Plan
1 *name* = “The Legend of Zelda: Ocarina of Time”
2 *player_perspective* = “first person”
3 *genres* = “arcade”
4 *platforms* = “Nintendo”
5 *has_mac_release* = “no”

biGRU AT (Random) If you like arcade first person games, but it is a Mac release for the Nintendo. It is The Legend of Zelda: Ocarina of Time.

biGRU AT (BGUP) If you’re looking for a first person arcade game you can play on Nintendo, try The Legend of Zelda: Ocarina of Time. It has a Mac release.

biGRU AT (NUP) The Legend of Zelda: Ocarina of Time is a first person arcade game for Nintendo, and it is not available on Mac.

biGRU AT+P (Random) If you’re looking for an arcade first person game, though it doesn’t have a Mac release or a Nintendo game is The Legend of Zelda: Ocarina of Time.

biGRU AT+P (BGUP) If you’re looking for a first person arcade game for Nintendo, try The Legend of Zelda: Ocarina of Time. It is not available on Mac.

biGRU AT+P (NUP) The Legend of Zelda: Ocarina of Time is a first person arcade game for Nintendo. It is not supported on Mac.

Transformer AT (Random) If you like arcade first person perspective games, then you should try the Nintendo video game called The Legend of Zelda: Ocarina of Time.

Transformer AT (BGUP) If you like first person arcade games, then you should try the Nintendo video game called The Legend of Zelda: Ocarina of Time.

Transformer AT (NUP) The Legend of Zelda: Ocarina of Time is a first person arcade game for Nintendo. It is a game that is not available for Mac.

Transformer AT+P (Random) If you like arcade first person games, then you should try the Nintendo video game called The Legend of Zelda: Ocarina of Time. It was released for Nintendo.

Transformer AT+P (BGUP) If you’re looking for a first person arcade game for the Nintendo, try The Legend of Zelda: Ocarina of Time. It’s not available for Mac.

Transformer AT+P (NUP) The Legend of Zelda: Ocarina of Time is a first person arcade game for the Nintendo. It is not available on Mac or Linux.

BART AT (Random) If you’re looking for an arcade game with a first person perspective that doesn’t have a Mac release, then you should try the Nintendo game The Legend of Zelda: Ocarina of Time.

BART AT (BGUP) If you like first person arcade games, then you should try the Nintendo video game called The Legend of Zelda: Ocarina of Time. Unfortunately, it doesn’t have a Mac release.

BART AT (NUP) The Legend of Zelda: Ocarina of Time is a first person arcade game for Nintendo, but it is not supported on Mac.

BART AT+P (Random) If you’re looking for an arcade game with a first person perspective that doesn’t have a Mac release, you should check out the Nintendo video game called The Legend of Zelda: Ocarina of Time.

BART AT+P (BGUP) If you’re looking for a first person arcade game for Nintendo consoles, check out The Legend of Zelda: Ocarina of Time. It’s not available on Mac though.

BART AT+P (NUP) The Legend of Zelda: Ocarina of Time is a first person arcade game for Nintendo consoles only. It is not available on Mac.

Random Utterance Plan

1	*genres* = “puzzle”
2	*genres* = “driving/racing”
3	*name* = “Metro 2033”
4	*genres* = “platformer”
5	*genres* = “music”

BGUP Utterance Plan

1	*genres* = “music”
2	*genres* = “platformer”
3	*genres* = “puzzle”
4	*name* = “Metro 2033”
5	*genres* = “driving/racing”

NUP Utterance Plan

1	*name* = “Metro 2033”
2	*genres* = “driving/racing”
3	*genres* = “platformer”
4	*genres* = “puzzle”
5	*genres* = “music”

biGRU AT (Random) If you like puzzle racing games, try Metro 2033. It is a platformer music game.

biGRU AT (BGUP) The music platformer puzzle game, Metro 2033, is a driving/racing game.

biGRU AT (NUP) Metro 2033 is a driving/racing platformer with puzzle elements. It’s a music game.

biGRU AT+P (Random) If you like puzzle games, Metro 2033 is a driving/racing platformer. It is a music game.

biGRU AT+P (BGUP) If you like music platformer games with puzzles, Metro 2033 is a driving/racing game.

biGRU AT+P (NUP) Metro 2033 is a driving/racing platformer with puzzle solving. It is a music game.

Transformer AT (Random) If you like puzzle games, Metro 2033 is a driving/racing platformer. It is a platformer game.

Transformer AT (BGUP) If you like music games, then you should try the puzzle platformer, try Metro 2033.

Transformer AT (NUP) Metro 2033 is a driving/racing platformer with puzzle solving.

Transformer AT+P (Random) If you are looking for a puzzle platformer, try Metro 2033. It is a driving/racing platformer game that is a music game.

Transformer AT+P (BGUP) If you like music games, then you should try Metro 2033. It’s a puzzle driving/racing game.

Transformer AT+P (NUP) Metro 2033 is a driving/racing platformer puzzle game.

BART AT (Random) If you’re looking for a puzzle driving/racing game, try Metro 2033. It’s a platformer with music elements.

BART AT (BGUP) If you’re looking for a music platformer with puzzle solving, Metro 2033 is a driving/racing game.

BART AT (NUP) Metro 2033 is a driving/racing platformer with puzzle and music elements.

BART AT+P (Random) If you’re looking for a puzzle game with driving/racing, Metro 2033 is a platformer with music elements.

BART AT+P (BGUP) If you’re looking for a music platformer with puzzle solving, Metro 2033 is a driving/racing game.

BART AT+P (NUP) Metro 2033 is a driving/racing, platformer, puzzle, music game.

E Human Evaluation Details

Two separate annotators ranked the 100 E2E outputs and another two annotators ranked the 100 ViGGO outputs. The annotators were either undergraduate or PhD students experienced in NLP research but not involved in the paper. Three were native English speakers and the fourth was a highly fluent English speaker. When computing Kendall's τ on E2E, three instances were not computable because one annotator gave all three outputs the same rank. These three instances were assigned $\tau = 0$ equivalent to no correlation.

Annotators were given the following instructions and then made their ranking annotations in Google Sheet:

Instructions: *You will be shown 3 utterances that are informing you about either a restaurant or a video game. Please rank the utterances according to their naturalness (i.e. fluency and/or degree to which you believe they were written by a native English speaker). 1 = most natural, 3 = least natural.*

Here is an example:

	Rank
(0) There is an English food place near the Sorrento with a price range of less than £20 called Blue Spice.	2
(1) Blue Spice serves English food for less than £20 and is located near the Sorrento.	1
(2) Serving English food near the Sorrento with a price range of less than £20 is Blue Spice.	3

Here I have decided that (1) feels the most natural, nicely breaking up information into a conjunction, while (2) seems least natural because of its run on gerund phrase in a copula. (0) is a little bit of a run on but not egregious.

Do not worry if one utterance does not have all the same or inconsistent facts as the others. Judge them only on their naturalness.

In many cases you will probably feel that two or more examples are equivalent in naturalness. In this case give them the same rank. E.g.,

	Rank
(0) There is a place that serves Japanese food in the riverside area near Café Sicilia called the Twenty Two.	1
(1) The Twenty Two serves Japanese food and is located near Café Sicilia in the riverside area.	1
(2) Serving Japanese food in the riverside area near Café Sicilia is the Twenty Two.	2

When making ties, make sure the next lowest rank follows numerically, i.e. if there is a tie for 1, the next lowest rank should be 2. In other words don't do this:

	Rank
(0) There is a place that serves Japanese food in the riverside area near Café Sicilia called the Twenty Two.	1
(1) The Twenty Two serves Japanese food and is located near Café Sicilia in the riverside area.	1
(2) Serving Japanese food in the riverside area near Café Sicilia is the Twenty Two.	3

You will annotate 100 sets of 3 utterances.