Appendices

A Diverse Outputs

We demonstrate the ability of the model to produce diverse outputs by showing examples of generation from graphs with 4, 5 or 6 edges. For each graph, we show every \( k \)th plan, where \( k \) is chosen so that our 25 examples cover the top 10% of the plans, and order them by the scores assigned to them by the scoring model (the score is shown to the right of each plan, as well as the rank in the list). Higher scoring plans correspond to more natural plans, according to our model, but all of them are viable options. Then, for each plan we show the corresponding text generated by the NMT model. This provides a glimpse of: (1) the quality of the scoring model; (2) the diversity of the plans; (3) the naturalness of the generation.

For the plans, color boxes indicate entities, and gray boxes around them indicate bracketing. Vertical bars indicate sentence splits. For the generated text, each entity is underlines with the color corresponding to its box.

A.1 Example: Graph with 4 Edges

Figure 5 shows a random 4-edge graph from the seen part of the test set. Figure 6 shows the plans and Figure 7 the corresponding texts.

![Diagram](image.png)

Figure 5: Example of a graph with 4 edges
Figure 6: 25 random linearized plans (out of 1,295 possible plans) for the graph in Figure 5, and their ranks and model scores.
Figure 7: Realizations of the plans from Figure 6 as produced by the NMT realizer.
A.2 Example: Graph with 5 Edges

Figure 8 shows a random 5-edge graph from the seen part of the test set. Figure 9 shows the plans and Figure 10 the corresponding texts.

Figure 8: Example of a graph with 5 edges
Figure 9: 25 random linearized plans (out of 9,460 possible plans) for the graph in Figure 8, and their ranks and model scores.
Figure 10: Realizations of the plans from Figure 9 as produced by the NMT realizer.
A.3 Example: Graph with 6 Edges

Figure 11 shows a random 6-edge graph from the seen part of the test set. Figure 12 shows the plans and Figure 13 the corresponding texts.
Figure 12: 25 random linearized plans (out of 171,024 possible plans) for the graph in Figure 11, and their ranks and model scores.
Figure 13: Realizations of the plans from Figure 12 as produced by the NMT realizer.
B Manual Evaluation Setup

When performing pairwise system comparisons, we show the user, for each set of RDFs, the two texts produced by the compared systems in random order, along with the RDF triplets in textual and image forms as a reference. For consistency, both texts are normalized by lower-casing and splitting tokens on punctuation. The same interface is used for turkers (for the fluency task) and local annotators (for the faithfulness task).

B.1 Fluency Evaluation by Crowd

We evaluate on the RDF sets in the original WebNLG manual evaluation setup. The task is performed by mechanical-turk workers. The workers are presented with the question:

“Which text reads more fluently?”

which can be answered by either Text 1, Text 2 or Both are equally good or bad.

We paid 0.08$ per hit, employing three workers on each. For qualification, workers were required to have over 98% hit approval rate, and over 1000 approved hits.

B.2 Faithfulness Evaluation by Expert

To obtain reliable fine-grained evaluation of semantic faithfulness, the first author annotated the system outputs of StrongNeural and BestPlan.

For each text, we present all the RDF input triplets, and ask the annotator to choose for each triplet one of three options: (1) This triplet is expressed in the text; (2) This triplet is not expressed in the text (ommitted); (3) The text expresses a relation between the two entities that is different than the one specified for them in the RDF triplet (wrong lexicalization). Also, for each text, we ask the annotator to count the number of facts that were wrongly over generated, counting both repeated facts and hallucinated ones.

C Training Parameters

For the realization model we use the OpenNMT toolkit (Klein et al., 2017) with pre-trained GloVe.6B word embeddings (Pennington et al., 2014), downloaded from http://nlp.stanford.edu/data/glove.6B.zip. We used the default parameters (except for the -copy_attn flag). This corresponds to the following values:

- train_steps = 40000
- save_checkpoint_steps = 2000
- batch_size = 16
- word_vec_size = 300
- layers = 3
- copy_attn
- position_encoding