Neural NLG for Methodius: From RST Meaning Representations to Texts*

Symon Jory Stevens-Guille[†] **Aleksandre Maskharashvili**[†] **Amy Isard**[‡]
Xintong Li[†] and **Michael White**[†]

[†]The Ohio State University [‡]University of Hamburg

Abstract

While classic NLG systems typically made use of hierarchically structured content plans that included discourse relations as central components, more recent neural approaches have mostly mapped simple, flat inputs to texts without representing discourse relations explicitly. In this paper, we investigate whether it is beneficial to include discourse relations in the input to neural data-to-text generators for texts where discourse relations play an important role. To do so, we reimplement the sentence planning and realization components of a classic NLG system, Methodius, using LSTM sequence-to-sequence (seq2seq) models. We find that although seq2seq models can learn to generate fluent and grammatical texts remarkably well with sufficiently representative Methodius training data, they cannot learn to correctly express Methodius's SIMILARITY and CONTRAST comparisons unless the corresponding RST relations are included in the inputs. Additionally, we experiment with using self-training and reverse model reranking to better handle train/test data mismatches, and find that while these methods help reduce content errors, it remains essential to include discourse relations in the input to obtain optimal performance.

1 Introduction

Traditional approaches to the task of natural language generation (NLG) have employed a pipeline of modules, moving from an initial abstract meaning representation (MR) to human-readable natural language (Reiter and Dale, 2000). In the last decade, the success of neural methods in other domains of natural language processing (NLP) has led to the development of neural 'end-to-end' (e2e) architectures in NLG (Dušek et al., 2020), where a direct mapping from MRs to text is learned. Since target texts for training neural models are typically crowd-sourced, the neural approach promises to make it easier to scale up the development of NLG systems in comparison to classic approaches, which generally require domain- or application-specific rules to be developed, even if the modules themselves are reusable.

Accompanying the increase in crowd-sourced corpora has been a comparative simplification of both MRs and tasks. In particular, classic NLG systems typically made use of hierarchically structured content plans that included discourse relations as central components, where the discourse relations — often based on Rhetorical Structure Theory (RST) (Mann and Thompson, 1988; Taboada and Mann, 2006) — group together and connect elementary propositions or messages (Hovy, 1993; Stede and Umbach, 1998; Isard, 2016). By contrast, more recent neural approaches — in particular, those developed for the E2E and WebNLG shared task challenges — have mostly mapped simple, flat inputs to texts without representing discourse relations explicitly.

The absence of discourse relations in work on neural NLG to date is somewhat understandable given that neural systems have primarily tackled texts that merely describe entities, rather than comparing them, situating them in time, discussing causal or other contingency relations among them, or constructing persuasive arguments about them, where discourse relations are crucial for coherence (Prasad et al., 2008). Recently, Balakrishnan et al. (2019a) have argued that discourse relations should be reintroduced into neural generation in order to enable the correct expression of these relations to be more reliably controlled. However, they do note that only 6% of the crowd-sourced E2E Challenge texts contain discourse connectives ex-

*The first two authors are listed in random order (equal contribution), then the other authors are listed in alphabetical order by last name.

E-mail: stevensguille.1@buckeyemail.osu.edu

Proceedings of The 13th International Conference on Natural Language Generation, pages 306–315,
Dublin, Ireland, 15-18 December, 2020.

pressing CONTRAST, and though they introduce a conversational weather dataset that uses both CONTRAST and JUSTIFY relations with greater frequency, it is fair to say that the use of hierarchical MRs that incorporate discourse relations remains far from common practice.

In this paper, we investigate whether it is beneficial to include discourse relations in the input to neural data-to-text generators for texts where discourse relations play an important role. To do so, we reimplement the sentence planning and realization components of a classic NLG system, Methodius (Isard, 2016), using LSTM sequence-to-sequence (seq2seq) models, since Methodius makes similarity or contrast comparisons in most of its outputs. Specifically, rather than crowd-source output texts for Methodius's content plans, we run the existing system to obtain target texts for training seq2seq models, and experiment with input MRs (derived from the content plans) that contain discourse relations as well as ones that leave them out.[1]

In our experiments, we observe that the seq2seq models learn to generate fluent and grammatical texts remarkably well. As such, we focus our evaluation on the correct and coherent expression of discourse relations. Since the Methodius texts are somewhat formulaic following delexicalization and entity anonymization, it is possible to write accurate automatic correctness checks for these relations. Using these automatic checks, we find that even with sufficiently representative Methodius training data, LSTM seq2seq models cannot learn to correctly express Methodius's similarity and contrast comparisons unless the corresponding RST relations are included in the inputs. This is an at least somewhat surprising result, since these relations are easily inferred from the input facts being compared.

The major conclusion of our experiments is that explicitly encoding discourse information using RST relations boosts coherence by enabling rhetorical structure to be reliably lexicalized. Several techniques for improving the models are also considered, especially for situations where the training data exhibits mismatches with the test data (as can happen in practice). One technique involves outputting a beam of possible text outputs and reranking them by checking the correspondence between the input meaning representation and the meaning representation produced by using a reversed model to map texts to meaning representations. The other technique is self-training (Li and White, 2020), i.e., using an initial model to generate additional training data. This method drastically increases the amount of training data available for what is otherwise quite a small corpus. The upshot of these techniques is moderate improvement in the performance of both models with respect to the evaluation metrics just mentioned. But the conclusion remains that the model trained on explicit RST information continues to outperform the model without explicit RST structure in the input.

[1]The data and code for this paper can be accessed by the following link: https://github.com/Methodius-Project/Neural-Methodius.

2 Methodius

The Methodius system (Isard, 2016) was developed for multilingual text generation, based on the M-PIRO project (Isard et al., 2003; Isard, 2007) which focused on museum exhibit descriptions. Methodius consists of several components. The content module selects content from a database and creates a content plan, which is a tree where the nodes are labeled with rhetorical relations or facts, following the structures proposed in RST. Fig. 1 shows a content plan. The content plan is rewritten into a sequence of logical forms, one per sentence, by the sentence planner. The logical forms are then realized as a text by means of a Combinatory Categorial Grammar (CCG) using OpenCCG (White, 2006).

The Methodius system is designed to respond to the behaviour of the its intended users. Sequences of exhibits, dubbed 'chains', are constructed while the user moves through the museum. The chains control dependencies between exhibit descriptions, limit redundancy, and provide discourse continuity.

While RST defines a number of rhetorical relations, Methodius incorporates only four of them: ELABORATION, JOINT, SIMILARITY and CONTRAST. ELABORATION connects the main fact about a focal entity with other, peripheral facts about that entity. JOINT connects two facts of equal status. SIMILARITY and CONTRAST each connect two facts of equal status, but they do opposite jobs: SIMILARITY is used to express the similarity of two entities in terms of a commonly shared feature, while CONTRAST is used to show that the values of a shared feature of the given entities differ. For instance, *unlike the previous coins you saw, which are located in the Athens Numismatic Museum, this*

```
[__content_plan
 [__rst_elaboration
  [__fact_type [__arg1 entity0 ] [__arg2 marriage_cauldron ] ]
  [__rst_joint [__fact_creation_period [__arg1 entity0 ] [__arg2 "classical period" ] ]
               [__fact_creation_time [__arg1 entity0 ] [__arg2 "between 420 and 410 B.C." ] ] ] ]
 [__rst_similarity
  [__fact_painting_technique_used compare_additive [__arg1 entity1 ]
                                                   [__arg2 "red figure technique" ] ]
  [__fact_painting_technique_used [__arg1 entity0 ] [__arg2 "red figure technique" ] ] ]
 [__optional_type [__arg1 entity1 ] [__arg2 vessel ] ] ]
```

Figure 1: Content plan corresponding to the text (1a)

tetradrachm is located in the National Museum of Athens – here *unlike* signals CONTRAST. In the following example, *like* signals SIMILARITY: *like the previous coins you saw, this tetradrachm is located in the National Museum of Athens*.

In the experiments discussed below we focus on SIMILARITY and CONTRAST because the Methodius corpus lexicalizes them. Due to the dynamic generation of the exhibit descriptions, SIMILARITY and CONTRAST link information in the current exhibit to previously mentioned exhibits and their properties—as such, correctly generating such expressions is vital to maintaining the coherence of the exhibit chain.

3 Data Preprocessing

3.1 Delexicalization

The textual output of Methodius is pseudo-English with some expressions replaced by canned text, the morpho-syntactic descriptions of which are not present in either the content plan or in the logical form. Instead the canned text is retrieved from the Methodius system's database by looking up the reference given in the content plan. Such canned texts might occur infrequently in a relatively small corpus. To avoid data sparsity, we substitute canned texts by their labels, cf. (1b), (1a). Note that the textual output of Methodius doesn't contain non-terminal symbols the sort used in Balakrishnan et al.'s approach. We use only *special terminal symbols*, which appear both in content plans (decorating terminal nodes in the tree) and in texts (representing the corresponding chunks of canned texts).

3.2 Anonymization and Augmentation

We anonymize exhibits by replacing them with entity0, entity1, etc in both the content plans and corresponding text. In each text, there is a single focal exhibit. The focal exhibit is compared to one or many exhibits and this is expressed in text using singular and plural forms respectively (e.g. *the other vessel, which originates from region1* VS *the other coins, which were created in city0*). We use two substitution forms: *entity1* (for singular) and *entityplural*.

Content plans are augmented with relevant information concerning the types of exhibits that occur in a content plan. The type predicate relates an exhibit to the NP it corresponds to in the text. This information is encoded within the Methodius logical form and thus is available for the Methodius system when it comes to generating text. However, since we anonymize exhibits and we ignore the logical forms, we need to explicitly provide the type information of each exhibit. Methodius sometimes produces content plans in which the first FACT__TYPE is missing arg2. This missing position corresponds to the focal exhibit in the text. The modified corpus regiments the input by ensuring every FACT__TYPE includes arg2. For every exhibit in the the Methodius content plan not explicitly typed we add a new OPTIONAL_TYPE branch to the tree which includes the type of the exhibit.

(1) a. This is a marriage cauldron and it was created during the classical period in between 420 and 410 B.C. Like the other vessel you recently saw, this marriage cauldron was decorated with the red figure technique.

b. this is a marriage cauldron and it was created during historical-period0 in exhibit0-creation-time . like the other vessel you recently saw , this marriage cauldron was decorated with painting-technique0 .

4 Data

Since one of our objectives is to compare the performance of neural networks on data with and without

Data	CONTRAST	SIMILARITY	Neither
train	840	2911	553
valid	79	292	51
standard test	166	495	138
challenge test	77	80	80

Table 1: Distribution of RST types in content plans in train and test data

Data	Average Words	Average Tokens
train	52	95
valid	52	96
standard test	40	73
challenge test	29	52

Table 2: Average numbers of tokens in content plans and average numbers of words in corresponding texts in train and test data

rhetorical relations, we call one of datasets RST and the other FACT. The model names follow the same convention. The distributions of RST types in content plans is shown in Table 1, where the first and second numbers correspond to the number of content plans including CONTRAST and SIMILARITY, respectively, while the third corresponds to the number of content plans which include neither of these RST types. The average lengths for input (number of tokens) and output (number of words) are shown in Table 2. The output of Methodius is limited with respect to both the homogeneity and lengths of the texts–Methodius only infrequently produces very short or long texts, e.g. one or six sentences respectively. One of the test sets, which is described below, is explicitly constructed to determine whether the model's knowledge of discourse structure is limited by the length of the texts it sees.

4.1 Training Set

In the training set, there are around 4300 examples harvested by using the Methodius system. The higher number of inputs with SIMILARITY (2911) is due to the Methodius system. This proportion of SIMILARITY persists into every split except the challenge test set, where the number of inputs of distinct RST types is more homogeneous.

4.2 Test sets

We have two splits of data for our experiments. One we dub the 'challenge split', the other the 'standard split'. The major difference between them is their average lengths. The average length of the challenge split items are roughly half the length of the training set items, while the average length of the standard split is roughly seventy five percent of the training set items.

Standard In the standard split, the average length of items in the training and validation sets is roughly the same; the distribution of lengths is similar in the training, valid, and test sets but the training set still includes slightly longer sequences on average. The proportion of items with distinct RST types is roughly the same between the train, valid, and standard test sets. This test set doesn't identify possible effects of item length on correct discourse structure production.

Challenge The challenge test set consists of items on average half the length of the the average lengths of items in the train and valid sets. Due to the lower frequency of short items produced by Methodius, the number of items in the challenge set is reduced. The distribution of items with CONTRAST and SIMILARITY is homogeneous.With respect to distinguishing RST types, the challenge test set is no more difficult than the standard test set; the item length is shorter but no less structured. Moreover, the set of lexemes–including delexicalized expressions– which occur in the test set are present in the training set. However, there are patterns in the test set which are uncommon or unseen in the training set, e.g. one content plan in the challenge set begins with CONTRAST but no such items are found in training. This distinguishes possible effects of length, e.g. 'RST type X occurs in the third sentence', from effect of RST tree structure in the input for correct discourse structure production, i.e. 'RST type X must correspond to lexeme/structure Y'. These challenge test-specific content plans help to determine how well a model learns to associate certain strings with either CONTRAST or SIMILARITY. If the model stumbles on shorter texts then its knowledge of RST structure might be (erroneously) conditioned on item length.

5 Evaluation Methods

5.1 Metrics on Special Terminals

Since the data we generate after preprocessing contains certain expressions which we dub 'special terminals', these expressions can be tracked between the target and the hypothesis. By obtaining metrics based on the correspondence between these special

terminals, we get a picture how close the hypothesis is to the target. This measure enjoys some useful properties. Firstly, it's cheap—it is defined solely in terms of expressions which occur both in the input (content plan) and in the output (text). Second, the special terminals stand for important parts of the text—those ones that are explicitly provided as values to features in the content plan (since they are terminals). Hence, having information about their presence gives us a good hint of the quality of a text.

In addition to standard evaluation metrics scores such as BLEU4, we report the following metrics for each test item:[2]

- Repetitions: A special terminal is present in the hypothesis n times but in the target text it occurs m times, where $m < n$. We calculate $n-m$ for every such special terminal and sum up.
- Omissions: How many times special terminals occurring in the target text are not generated at all in the hypothesis.
- Hallucinations: Number of occurrences of those special terminals in the hypothesis that have no occurrence in the target.

5.2 Discourse adverbials for Contrast and Similarity

We also provide a count for the number of items in which (within tables in Appendix A): (a) Target and Hypothesis both contain *unlike*; (b) Target and Hypothesis both contain *like*; (c) Target contains *unlike* but Hypothesis generates *like*; (d) Target contains *like* but Hypothesis generates *unlike* (Like vs. Unlike); (e) Target contains neither *like* nor *like* and the same holds of Hypothesis (No rel in both); (f) the rest of the cases.

6 Self-training

With most NLG applications, large amounts of parallel data are not readily available. This is true even in the case of Methodius, because there are a finite number of exhibits and facts and thus the number of meaningful combinations which can be constructed from them is limited. In order to reduce annotated data needs, Kedzie and McKeown (2019), Qader et al. (2019) and He et al. (2020) propose self-training methods for NLG. Li and White (2020) explore self-training for the more challenging case of generating from compositional input representations. Self-training involves the construction of unlabeled data. The process of self-training is the following. First, the model is trained on the initial parallel data, i.e. the data used in the models without self-training. Subsequently, an additional set of unlabeled inputs is provided: such data might exist but be unlabeled but if no such data exists it can be generated (e.g., handcrafted using some heuristics). The unlabelled inputs are, in the present context, content plans without corresponding output text. Next the existing model is used to generate the labels for the unlabeled data. This procedure results in a new set of parallel data. Because its labels don't come from the data—since they're outputs of the model—this cannot be considered parallel data in the full sense. We dub the resulting data 'pseudo-labelled.' We train a new model on this data. Then we reuse the genuine parallel data for fine tuning this model. This process can be repeated to generate various models. (1) describes the process in brief:

Algorithm 1: Vanilla Self-Training

```
Train a model on ℒ;
repeat
    Pseudo-label the unlabeled data in 𝒰;
    Train a model on the pseudo-parallel
      data;
    Fine-tune the model on ℒ;
until convergence or maximum iteration;
```

7 Reranking with reverse models

In the syntax-semantics interface, the parsing task is usually to build a correct semantic (or syntactic) representation of a sentence. One can consider this task with respect to neural networks—which operate on sequences—straightforwardly by reversing the order of the parallel data: the source sequence (meaning) becomes the target, and the target sequence (text) becomes the source. Following the terminology of Li and White (2020), we call such models reverse models, while models that generate text from meaning representations are forward models.[3]

[2] The term 'hypothesis' is used for the output of the model, following the terminology used in Fairseq.

[3] While this is an arbitrary choice of terminology, in the context of NLG it seems to be appropriate to call the forward model the one that generates text out of meaning representation.

We can rerank the output of a forward model with the help of its corresponding reverse model. Given several outputs of a beam search of the forward model, we select the one that makes the *best* meaning representation if it is given to a reverse model as an input. Here, best means the one that has lowest perplexity with respect to forced decoding. One can combine self-training and reranking: Train forward and reverse models on the parallel data and then train forward and reverse models on the pseudo-parallel data. Afterwards fine-tune them again on the initial parallel data. Subsequently, use the reverse models to rerank the output of the forward models.

Algorithm 2: Reverse Model Reranking

```
Train forward and reverse models on L;
repeat
    Pseudo-label the unlabeled data in U
      with reverse model reranking;
    Train forward and reverse models on the
      pseudo-parallel data;
    Fine-tune both models on L;
until convergence or maximum iteration;
```

8 Experiments and Results

We ran self-training experiments with two sets of unlabeled data. One of them consists of the content plans generated by Methodius. The other one, dubbed 'heuristic,' is developed from the existing labeled data. The heuristic data is produced by the following method: for every content plan produced by Methodius, extract the set of subtrees of the content plan which respect some *soft* constraints on structure. We avoid extracting trees that start with an optional type. The subtrees are randomly selected but their distribution is required to closely follow the distribution of distinct RST types in the training data.

Since the size of the Methodius data set is limited, the heuristic data set provides useful cheap supplementary content for training (compared to the cost of eliciting text corresponding to content plans through e.g. Turkers). We are thus interested whether having genuine Methodius content plans, which are not straightforward to generate in large amounts, could be completed by a heuristic data set generated from the labeled training data set.

The FACT models were trained on the FACT versions of the data set, which is obtained by simply deleting the RST structure from the RST data set.[4]

We refer to the models (for sake of clarity) by the names in Table 3.

Model Name	Self-training data size	Generated by
RST-SM	947	Methodius
FACT-SM	934	Methodius
RST-LG	81970	heuristic
FACT-LG	76531	heuristic

Table 3: Models trained on training set of size 4304

There are only 947 content plans for self-training, while the training set size is 4304. The limited number of content plans for self-trainining is due to the homogeneity of the Methodius output, the intention to sync the length of training and test sets, and the finite number of exhibits in the Methodius data base. These content plans, which are harvested from Methodius, are on average just half the length of the content plans in the training set. Their shortness ensures the system is exposed to items of multiple lengths. Because of their reduced length and their production by the Methodius system, variation in the content of the short sequences is limited. The unique unlabelled data size differs between RST and FACT data sets, because the data for FACT is produced by pruning the RST data, the deletion of structure reduces the heterogeneity of data, resulting in fewer unique sequences for the FACT-LG input.

We trained the following models:

- LBL: A standard LSTM seq2seq model with attention on the labeled data, which is also the base model for the other methods.
- ST-VAN: A model trained with vanilla self-training.
- ST-RMR: A model self-trained with reverse model reranking for pseudo-labeling.

Models were trained over several iterations, though for exposition the results reported below concern just the best model iterations.[5]

BLEU4 is calculated on both the standard and challenge test sets. BLEU4, though limited in the conclusions it supports, seems informative enough to allow one to distinguish between RST and FACT

[4] Deleting RST structure results in the deletion of the tree structure too.

[5] In addition to LSTM models, we trained a baseline transformer on the labeled data but the results were unsatisfactory.

models; we report it in Appendix D. BLEU4 is on average 5 or more points higher for RST models than FACT models across the test sets.

8.1 Repetitions, Hallucinations and Omissions

We count the sum of repetitions, hallucinations and omissions per test set and report the average per item, simply dividing the sum by the number of test set samples.

Fig. 2 and Fig. 3 show the results, chiefly the uniform improvement of the self-training and reranking models over the baseline LSTM models.

RST-SM with self-training is the best model. RST-SM with both self-training and reverse model reranking produced some of the best results too.

RST-SM and RST-LG show similar performance when it comes to repetitions, hallucinations, and omissions on the standard test set. RST-SM outperforms RST-LG on the challenge set. RST models uniformly outperform FACT models.

We observed the models sometimes produced stuttering, i.e. multiple repetition. Even one of the best models with respect to the standard test set—RST-SM-ST-VAN (see Fig. 2)—produced two examples of stuttering (out of 799) with 57 and 59 repetitions respectively. Just these two outputs nearly doubled the average error rate of RST-SM-ST-VAN. The other models reported here did not produce such extreme stuttering. But despite stuttering, RST-SM-ST-VAN is still the best model with respect to the metrics considered here. In Appendix C, model performance is reported by simply counting the total number of test examples in which a model generates neither repetitions, nor omissions, nor hallucinations.

The following error from FACT-LG-ST-RMR shows multiple hallucination of the exhibit item's creation time.

T this is an imperial portrait and it portrays roman-emperor0 . like the coin you recently saw , this imperial portrait was created during historical-period0 .

H this is an imperial portrait and it portrays roman-emperor0 . like the coin , this imperial portrait was created during historical-period0 . it was created in entity0-creation-time and it was created in entity0-creation-time .

Further errors are shown in Appendix E.

8.2 Rhetorical Relation Generation

When the FACT-LBL model makes mistakes, such mistakes frequently correspond to the substitution of one lexeme marking a rhetorical relation for another marking a distinct (sometimes opposite) relation. The following hypothesis replaces the CONTRAST in the target with a SIMILARITY, misidentifying the origin of some previous exhibit in the chain.

T unlike the other exhibits you recently saw , which originate from region0 , this coin was originally from city0 .

H like the other exhibits you recently saw , this coin originates from city0 .

In the following hypothesis the erroneous substitution of SIMILARITY by CONTRAST leads to an outright contradiction:

T like the other exhibits you recently saw , this marriage cauldron is currently in museum0 .

H unlike the other exhibits you recently saw , which are located in museum0 , this marriage cauldron is located in museum0 .

Less frequently the insertion of SIMILARITY or CONTRAST compares the topic of an exhibit to itself:

T this is a statue and it was created during historical-period0 in entity0-creation-time .

H this is a statue and it was created during historical-period0 in entity0-creation-time . like the statue , this statue was created during historical-period0 .

The details of the number of errors and successes in generating discourse connectives are reported in Appendix A.

Fig. 4 and Fig. 5 show Fisher's Exact Test statistics for best performing RST (SM and LG) and FACT (SM and LG) models.

8.2.1 Standard Test

The best performances are shown by RST-SM and RST-LG. Even RST-LBL produces only 12 mistakes out of 799 test items. Production of rhetorical connectives corresponding to CONTRAST and SIMILARITY is uniformly correct. After fine tuning and reranking, the errors reduced to 0 and 2 respectively. With respect to the FACT models, LBL

makes mistakes, but improves upon self-training and reranking. Nonetheless RST models outperform the FACT models. While the best FACT model performs well with respect to producing the correct discourse connective/structure, this model produces serious content errors that render some outputs (discussed in Section 8.1) incoherent.

8.2.2 Challenge Test

On the challenge test, no model achieved perfect accuracy. The best performances are by RST-SM and RST-LG. Their performance is similar. It is worth noting that in the case of FACT-SM, reranking with self-training gave results comparable to RST-SM (there is no significance difference in terms of Fisher's test with significance at 5%). This is not the case for FACT-LG and RST-LG models. RST-LG-ST-RMR outperforms the best FACT-LG model (see Fig. 5).

From these experiments, we see that on the standard test set, RST-Large and RST-Small models performed best in terms of producing the correct discourse connective for SIMILARITY (respectively CONTRAST). While errors occurred—sometimes matching the results of the corresponding FACT models—RST models correctly distinguish between producing the lexeme for SIMILARITY versus CONTRAST, while FACT models sometimes confuse SIMILARITY with CONTRAST.

On the challenge data every model made errors. The RST models outperformed the corresponding FACT models, significantly in the case of RST-LG over RST-LG, as seen in Fig. 5.

Though the RST models yielded less dramatic improvements on comparisons in the challenge set, it is worth emphasizing that the RST models produce significantly fewer repetitions, omissions and hallucination compared to the FACT models (Figs. 6 and 7, Appendix C), further supporting the conclusion that the RST input produces better output. This result is interesting, since the content plans in the FACT models are shorter than those in RST models, yet still prompt the former models to produce more words than RST models do.

9 Related and Future Work

While traditional natural language generation systems, e.g. Methodius, often employ knowledge graphs, the use of such structure in neural NLG is underdeveloped. An exception in this respect is WebNLG (Gardent et al., 2017), which is a multilingual corpus for natural language generation. An

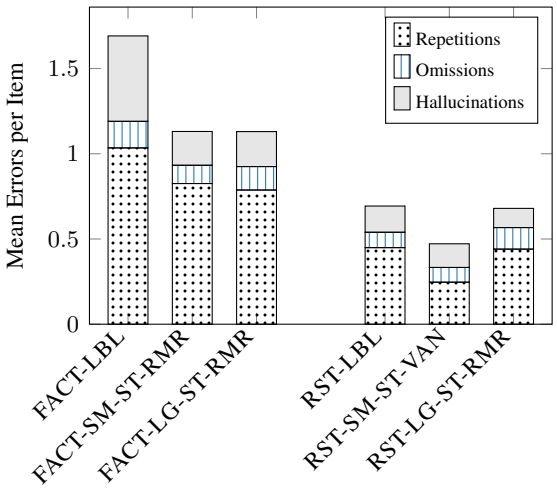

Figure 2: Standard Set

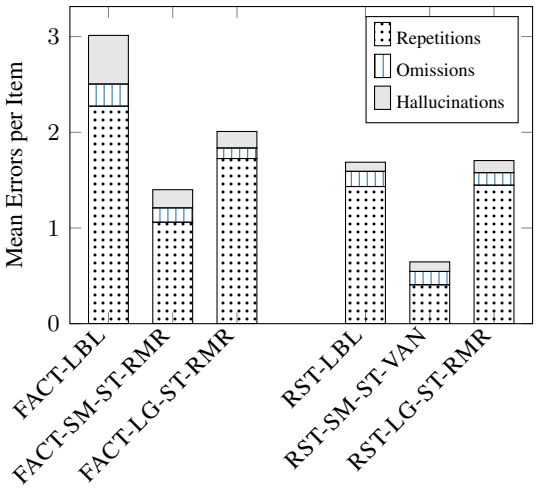

Figure 3: Challenge Set

entry of WebNLG is a set of RDF triples (representing *subj*, *predicate*, *object*) paired with the corresponding text, which is the sequences of sentences which serve as verbalization of those triples. But it is noteworthy that the main focus in WebNLG is micro-planning (sentence-level generation). Consequently, WebNLG only makes use of a single, implicit rhetorical relation, namely ELABORATION. ELABORATION is frequent in the Methodius corpus. But Methodius uses more interesting rhetorical relations, too, including CONTRAST and SIMILARITY, thus the content (both in terms of meaning representations and texts) is significantly different from WebNLG.

For future work, there are number of direction we intend to explore, including the following:

- Study whether large-scale pretrained models likewise fail to generalize well without dis-

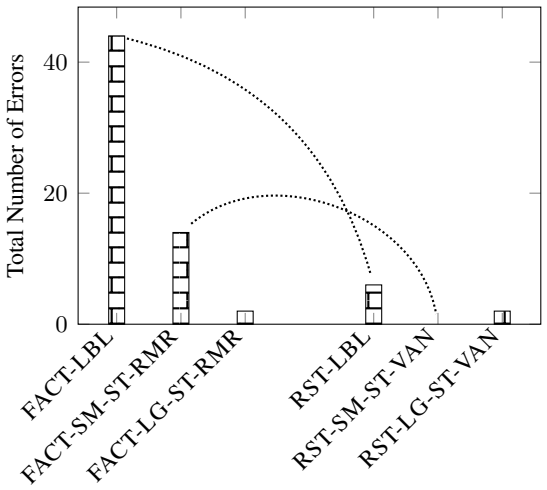

Figure 4: Standard Set: Errors in generating discourse cue words for SIMILARITY and/or CONTRAST (*unlike* and/or *like*), where towards errors counts if either there is an incorrectly generated discourse cue word, or there has been a cue word generated while the target has none, or no cue word is generated but the reference contains one. The dotted line links two models if there is a significant difference between their performance in terms of Fisher's Exact Test statistics (we take the significance threshold 5%).

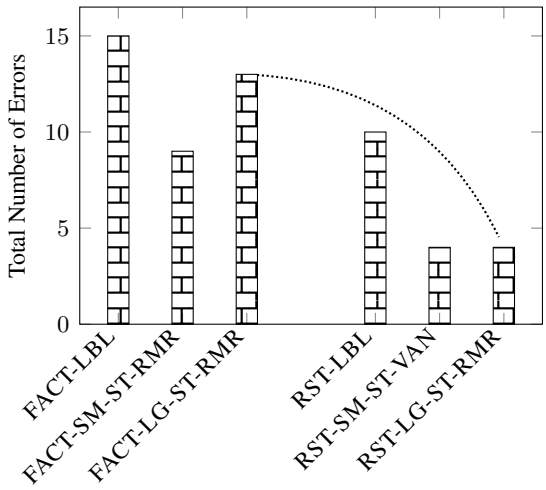

Figure 5: Challenge Set: Errors in generating discourse cue words for SIMILARITY and/or CONTRAST (*unlike* and/or *like*), where an item produces an error if either there is an incorrectly generated discourse cue word, or there has been a cue word generated while the target has none, or no cue word is generated but the reference contains one. The dotted line links two models if there is a significant difference between their performance in terms of Fisher's Exact Test statistics (with significance threshold of 5%).

course relations in the input.

- Experiment with more diverse outputs for Methodius, e.g. crowd-sourcing further outputs to express the content plans.
- Study whether constrained decoding could be used to reduce discourse structure errors.

10 Conclusions

The overall conclusion is that including RST relations in the input content plans is necessary to achieve optimum performance in correctly and coherently expressing discourse relations in the neural reimplementation of Methodius. This is somewhat surprising since the FACT-only inputs actually have all the information necessary to infer that a SIMILARITY or CONTRAST relation should be expressed, but the models nevertheless struggle to learn the desired same/different generalization. Moreover, the errors are often jarring–they produce genuine incoherence in the text.

We see the best performance from the RST model with small but clean self-training data (RST-SM), as it comes from Methodius and thus follows the same general patterns as the ones in the test set. The large RST model (RST-LG) had similar performance to the small one. FACT models, both small and large, show significant self-training improvements when reranking with reverse models. Because the RST baseline already performs relatively well, such an improvement is not observable with them. RST-SM with vanilla self-training already showed high performance. In the case of the FACT models, we saw that reranking with reverse models lowers repetitions, omissions and hallucinations in total. It was also beneficial for the RST-LG model.

Despite the highly regular nature of the rule-based texts, even our best models do not get close to zero content errors, highlighting the importance of continued work on eliminating these errors, e.g. using pretrained models (Kale, 2020; Kale and Rastogi, Forthcoming) or constrained decoding (Balakrishnan et al., 2019b).

Acknowledgments

This research was supported by a collaborative open science research agreement between Facebook and The Ohio State University.

References

Anusha Balakrishnan, Vera Demberg, Chandra Khatri, Abhinav Rastogi, Donia Scott, Marilyn Walker, and Michael White. 2019a. Proceedings of the 1st workshop on discourse structure in neural nlg. In *Proceedings of the 1st Workshop on Discourse Structure in Neural NLG*.

Anusha Balakrishnan, Jinfeng Rao, Kartikeya Upasani, Michael White, and Rajen Subba. 2019b. 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.

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.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada. Association for Computational Linguistics.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial Intelligence*, 63(1):341 – 385.

Amy Isard. 2007. Choosing the best comparison under the circumstances. In *Proceedings of the International Workshop on Personalization Enhanced Access to Cultural Heritage (PATCH07)*, Corfu, Greece.

Amy Isard. 2016. The methodius corpus of rhetorical discourse structures and generated texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1732–1736, Portorož, Slovenia. European Language Resources Association (ELRA).

Amy Isard, Jon Oberlander, Colin Matheson, and Ion Androutsopoulos. 2003. Speaking the users' languages. *Intelligent Systems, IEEE*, 18(1):40–45.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks.

Mihir Kale and Abhinav Rastogi. Forthcoming. Text-to-text pre-training for data-to-text tasks.

Chris Kedzie and Kathleen McKeown. 2019. A good sample is hard to find: Noise injection sampling and self-training for neural language generation models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 584–593, Tokyo, Japan. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xintong Li and Michael White. 2020. Self-training for compositional neural NLG. The third annual West Coast NLP Summit (WeCNLP), poster session.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical structure theory: Toward a functional theory of text organization. *Text & Talk*, 8(3):243 – 281.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. 2008. The penn discourse treebank 2.0. In *LREC*. Citeseer.

Raheel Qader, François Portet, and Cyril Labbé. 2019. Semi-supervised neural text generation by joint learning of natural language generation and natural language understanding models. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 552–562, Tokyo, Japan. Association for Computational Linguistics.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge university press.

Manfred Stede and Carla Umbach. 1998. Dimlex: A lexicon of discorse markers for text generation and understanding. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 1238–1242.

Maite Taboada and William C. Mann. 2006. Rhetorical structure theory: looking back and moving ahead. *Discourse Studies*, 8(3):423–459.

Michael White. 2006. Efficient realization of coordinate structures in combinatory categorial grammar. *Research on Language and Computation*, 4(1):39–75.