Self-Training for Compositional Neural NLG in Task-Oriented Dialogue

Xintong Li, Symon Jory Stevens-Guille, Aleksandre Maskharashvili and Michael White

Department of Linguistics
The Ohio State University
znculee@gmail.com stevensguille.1@osu.edu
maskharashvili.1@osu.edu mwhite@ling.osu.edu

Abstract

Neural approaches to natural language generation in task-oriented dialogue have typically required large amounts of annotated training data to achieve satisfactory performance, especially when generating from compositional inputs. To address this issue, we show that self-training enhanced with constrained decoding yields large gains in data efficiency on a conversational weather dataset that employs compositional meaning representations. In particular, our experiments indicate that self-training with constrained decoding can enable sequence-to-sequence models to achieve satisfactory quality using vanilla decoding with five to ten times less data than with ordinary supervised baseline; moreover, by leveraging pretrained models, data efficiency can be increased further to fifty times. We confirm the main automatic results with human evaluations and show that they extend to an enhanced, compositional version of the E2E dataset. The end result is an approach that makes it possible to achieve acceptable performance on compositional NLG tasks using hundreds rather than tens of thousands of training samples.

1 Introduction

Neural approaches to natural language generation (NLG) have received increasing attention due to their flexibility and end-to-end trainability (Wen et al., 2016; Mei et al., 2016; Dušek and Jurcicek, 2016; Dušek et al., 2019). However, despite using simplistic input meaning representations (MR), most neural models require large quantities of clean annotated training data in order to obtain good performance. As such, the time and expense required to obtain sufficient training data is a significant obstacle to deploying neural NLG models at scale.

To enable richer task-oriented dialogue, Balakrishnan et al. (2019) argue for using compositional, tree-structured MRs that include discourse relations, emphasizing the need for applications to exert control over these relations when generating text. Perhaps not surprisingly, their compositional input MRs further exacerbate annotated data needs. To address this issue, Balakrishnan et al. (2019) introduce a novel constrained decoding technique that nearly always yields correct output even in challenging cases. However, their constrained decoding method incurs a substantial runtime cost, making it too slow to deploy in task-oriented dialogue systems where low latency is a priority. Thus, finding ways to improve data efficiency for training models that perform satisfactorily with vanilla decoding remains an important challenge.

In order to reduce annotated data needs, Kedzie and McKeown (2019) and Qader et al. (2019) propose self-training methods for NLG, though they do not explore self-training for the more challenging case of generating from compositional input representations. Arun et al. (2020) do explore self-training with compositional inputs, but they do not consider constrained decoding. In this paper, we investigate for the first time whether constrained decoding can be used during self-training to enhance data efficiency for compositional neural NLG, since the speed of constrained decoding is much less of a concern during self-training than it is at runtime in dialogue systems. In particular, we adapt and extend He et al.'s (2020) approach to self-training for MT to the setting of neural NLG from compositional MRs, comparing vanilla self-training to self-training enhanced with constrained decoding as well as with reverse model reranking (Shen et al., 2019; Yee et al., 2019), a simpler technique where the n-best outputs of the forward model are reranked using scores from a reverse model. In both cases, the idea is to enhance the quality of the pseudo-annotated texts created during self-training, so that self-training can more successfully avoid entrenching the model's own

Query	**Context**	**MR**
When will it snow next?	Reference date: 29th September 2018	[CONTRAST [INFORM [LOCATION [CITY Parker]] [CONDITION_NOT snow] [DATE_TIME [DAY 29] [MONTH September] [YEAR 2018]]] [INFORM [DATE_TIME [DAY 29] [MONTH September] [YEAR 2018]] [LOCATION [CITY Parker]] [PRECIP_CHANCE_SUMMARY very likely] [CONDITION heavy rain showers] [CLOUD_COVERAGE partly cloudy]]]

Annotated Response
[CONTRAST [INFORM [LOCATION [CITY Parker]] is not expecting any [CONDITION_NOT snow]] , but [INFORM [DATE_TIME [COLLOQUIAL today]] there's a [PRECIP_CHANCE_SUMMARY very likely chance] of [CONDITION heavy rain showers] and it'll be [CLOUD_COVERAGE partly cloudy]]]

Table 1: Example compositional MR and annotated response from Balakrishnan et al.'s (2019) conversational weather dataset. In the actual dataset, discourse relations have a DS prefix (e.g., DS_CONTRAST), dialog acts have a DG prefix (e.g, DG_INFORM) and arguments have an ARG prefix (e.g., ARG_CITY); these are elided here for brevity.

mistakes. We show that self-training benefits considerably from both methods, and that constrained decoding yields especially large gains in data efficiency. In particular, our experiments indicate that using constrained decoding during self-training, rather than at runtime, enables standard sequence-to-sequence (seq2seq) models to achieve satisfactory quality with much reduced latency.

Our contributions are two-fold. On Balakrishnan et al.'s (2019) conversational weather dataset, we show that using constrained decoding during self-training and their SEQ2SEQ-TREE model at runtime yields comparable performance with 20% of the annotated data as using the full training set in supervised fashion, and by leveraging pretrained models, annotated data needs can be further reduced to 2%. We then confirm the main automatic metric results with human evaluations and show that they hold for Balakrishnan et al.'s (2019) enhanced version of the E2E dataset (Dušek et al., 2019).

2 Method

Neural NLG seq2seq models aim to generate a natural language text $\mathbf{y} = \langle y_1, \cdots, y_{|\mathbf{y}|}\rangle$ from a meaning representation $\mathbf{x} = \langle x_1, \cdots, x_{|\mathbf{x}|}\rangle$ by modeling the conditional probability

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{|\mathbf{y}|} P(y_i|\mathbf{y}_{<i}, \mathbf{x}), \qquad (1)$$

where $\mathbf{y}_{<i} = \langle y_1, \ldots, y_{i-1}\rangle$ denotes a prefix of $\mathbf{y}$ with length $i-1$. Usually, the model parameters are learned in supervised fashion from a set of annotated data $\mathcal{L} = \{\mathbf{x}_k, \mathbf{y}_k\}_{k=1}^{|\mathcal{L}|}$.

2.1 Compositional Inputs

Balakrishnan et al. (2019) propose to generate annotated responses from compositional, tree-structured MRs, as shown in Table 1. They demonstrate that compositional MRs offer greater control over the expression of CONTRAST and JUSTIFICATION discourse relations and lead to improvements in semantic correctness in a human evaluation, which they argue is important for conversational systems where external knowledge like user models may inform decisions around contrast, grouping, or justifications (Carenini and Moore, 2006; Walker et al., 2007; White et al., 2010; Demberg et al., 2011). By serializing the trees as shown, it is possible to use standard seq2seq models to effectively accomplish tree-to-tree generation. At runtime, the bracketing tokens can be straightforwardly removed to produce the final outputs.

2.2 Vanilla Self-Training

Hiring annotators to produce large amounts of clean, parallel data is costly, but it is often possible to automatically obtain lots of unlabeled data $\mathcal{U} = \{\mathbf{x}_l\}_{l=1}^{|\mathcal{U}|}$. To take advantage of the large unlabeled data $\mathcal{U}$, we adapt and extend He et al.'s (2020) semi-supervised self-training strategy, which has been successfully applied to MT. As shown in Algorithm 1, vanilla self-training starts from a base model trained with annotated parallel data $\mathcal{L}$, then (i) iteratively applies the current model to pseudo-label the unlabeled data with its predictions, (ii) trains a new model on the pseudo-labeled data, and (iii) fine-tunes the model on $\mathcal{L}$. Naturally, higher-quality pseudo-labeling can be expected to lead to more effective self-training by helping the model to avoid entrenching its own mistakes; below, we consider two strategies for improving generation during the pseudo-labeling step.

Algorithm 1: Vanilla Self-Training

```
1 Train a model on $\mathcal{L}$;
2 repeat
3     Pseudo-label the unlabeled data in $\mathcal{U}$;
4     Train a model on the pseudo-parallel
        data;
5     Fine-tune the model on $\mathcal{L}$;
6 until convergence or maximum iteration;
```

2.3 Constrained Decoding

Balakrishnan et al. (2019) demonstrate that constrained decoding can enhance the correctness of text generated with seq2seq models. In our experiments, we make use of an enhanced version of their constrained decoding method, both in the pseudo-labeling step of self-training as well as during runtime prediction.

Balakrishnan et al.'s (2019) constrained decoding method begins by scanning the input MR tree to build constraints on coverage and ellipsis.[1] During decoding, the non-terminals in the incrementally generated candidates are checked against the input tree for validity, where an output tree (ignoring terminals) is considered valid if it is isomorphic to the input tree up to sibling order and elided arguments. After each time step of the beam search, invalid candidates are filtered out to prevent hallucinations of tree structure, and closing brackets can only be generated when the non-terminals in the current subtree have all been covered. For example, in decoding a response for the MR in Table 1, if the prediction has followed the annotated response up until *and it'll be*, then a closing bracket cannot be generated at this point because the second INFORM is not complete, and CLOUD_COVERAGE is the only non-terminal that can be validly generated here.

[1]Arguments appearing multiple times in the input MR are only required to appear once in the output.

A problem with this post-filtering method of constrained decoding is that it can end up filtering out all candidates in the beam search, making it impossible for the decoding to proceed forward. To avoid this issue, we instead make use of a pre-filtering constraint. Specifically, rather than checking the non-terminals in $\mathbf{y}_{\leqslant i}$ after generating the next token in each time step i, our pre-filtering method instead determines all non-terminals that can appear as valid next tokens with $\mathbf{y}_{<i}$, then masks out all invalid non-terminals from the vocabulary before the next decoding step (the closing bracket is treated similarly). This ensures that all candidates in the beam are valid.

Another problem with Balakrishnan et al. (2019)'s constrained decoding is that it only constrains the generation of non-terminals. The generated terminals may be inconsistent with their parent argument non-terminals, even when placeholder terminals are used for delexicalized arguments. For example, a placeholder for city name should only be valid to generate inside an [ARG_CITY] argument instead of [ARG_DAY]. This kind of error is not common when the training data is sufficient, but it can severely harm the generation quality in data sparse situations. Therefore, in our enhanced constrained decoding, we constrain the generation of arguments by only nominating correspondingly valid placeholder terminals given a particular parent argument non-terminal.

While constrained decoding ensures the correctness of the partial tree structure and helps avoid inappropriate argument realizations, it does not constrain most terminals (i.e., the words themselves). As such, when the model ends up in a poorly trained part of its distribution, it can still hallucinate terminals; in particular, it can end up stuttering words until the maximum output length is reached, yielding an invalid tree structure. In these failure cases, we replace the output with the result of vanilla decoding, whose text is usually much better.

2.4 Reverse Model Reranking

As an alternative to constrained decoding's hard constraints on non-terminals, we also investigate a soft approach to favoring generated texts that correctly express the input MRs (Shen et al., 2019; Yee et al., 2019). To score the correctness of a generated text (with non-terminals removed), we train a reverse (i.e., parsing) model to generate a meaning representation $\mathbf{x}$ from a natural language text $\mathbf{y}$. Then, following beam search, the n-best generated texts are reranked with the forced decoding perplexity of the reverse model. When using reverse model reranking in self-training, the reverse model is also self-trained as shown in Algorithm 2.

```
Algorithm 2: Reverse Model Reranking
1 Train forward and reverse models on $\mathcal{L}$;
2 repeat
3     Pseudo-label the unlabeled data in $\mathcal{U}$
        with reverse model reranking;
4     Train forward and reverse models on the
        pseudo-parallel data;
5     Fine-tune both models on $\mathcal{L}$;
6 until convergence or maximum iteration;
```

3 Experiments

3.1 Setup

Datasets We conduct experiments on the publicly available conversational weather and enriched E2E datasets from Balakrishnan et al. (2019), focusing on the more challenging weather dataset. The weather task consists of 25k parallel items for training, and 3k for both validation and test. In the weather task, there are 1.6k unique tokens in the MRs, and 1.3k in the annotated responses. The enriched E2E dataset contains Balakrishnan et al.'s (2019) automatic enhancements to the E2E texts and MRs to include CONTRAST and JUSTIFICATION relations as well as slot-level annotations. The E2E task consists of 42k items for training, and 4.6k for both validation and test. In the E2E task, there are 60 unique tokens in the MRs, and 2.9k in the annotated responses. All the results are reported on the test set in the following experiments.

Unlabeled MR Creation For many NLG applications, unlabeled MRs can be generated in nearly unlimited quantities with a simulator, but unfortunately, we do not have access to the MR simulators for these two datasets. Our workaround is to create unlabeled MRs by modifying the MRs we have in the parallel data. Because there are contextual dependencies in the MRs, it would be hard to get realistic MRs just by sampling elements. Therefore, we instead delete all possible combinations of removable subtrees from the MRs in order to keep the pruned MRs meaningful. The removable subtrees are defined as an unprotected DG_INFORM or ARG that has at least one unprotected sibling, where protected elements are those that are manually identified as establishing context (e.g., ARG_LOCATION) or are children of CONTRAST and JUSTIFICATION relations, which have coherence-related contextual dependencies. In this way, we created 137k unlabeled MRs for weather and 143k MRs for E2E. When training a new model on pseudo-labeled data, we split 3k from each of them as validation data.

Models We report results for the following four kinds of models, where *-n means the method only uses $n\%$ of the parallel data from the full training set (three iterations of self-training were used, unless otherwise specified):

- LBL-n: A seq2seq model (LSTM with attention or BART), which is also the base model for the other methods
- ST-VAN-n: A model trained with vanilla self-training
- ST-CD-n: A model self-trained with constrained decoding for pseudo-labeling
- ST-RMR-n: A model self-trained with reverse model reranking for pseudo-labeling

Metrics We report the automatic metrics listed below on the raw model predictions, which have delexicalized fields (e.g., ARG_CITY). Nonterminal annotations are stripped when calculating BLEU-4 and auto–tree accuracy.

- BLEU-4 (Papineni et al., 2002): The BLEU evaluation from e2e-metrics (Dušek et al., 2018).
- Tree accuracy (Balakrishnan et al., 2019): The ratio of annotated responses that pass the validity constraints specified by the input MR. Note that if constrained decoding terminates successfully, it is guaranteed to pass the tree accuracy check, but vanilla decoding comes with no such guarantee.
- Auto–tree accuracy: Tree accuracy after using a reverse model (trained on all the paired data) to parse the text. Note that parse errors make auto–tree accuracy less accurate than tree accuracy, but this method can be used with plain text output.

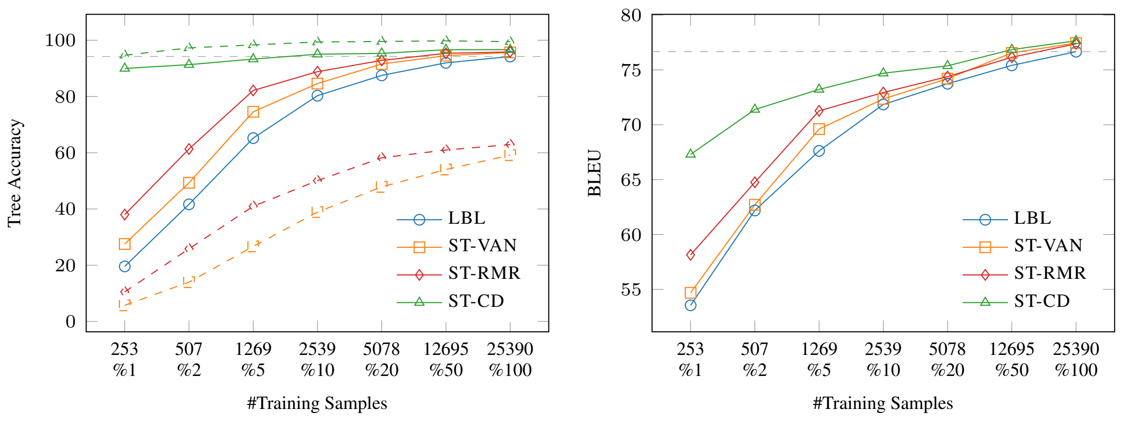

Figure 1: Tree accuracy and BLEU scores of the LSTM base model and three self-training strategies by parallel training data size with vanilla decoding on the conversational weather dataset. Tree accuracy on pseudo-labeled data is indicated by the same color dashed lines. Performance of the supervised model (LBL) using all of the labeled data is indicated by the gray dashed lines.

Implementation Our implementation[2] of self-training, constrained decoding and reverse model reranking is based on the same one-layer LSTM with attention approach as in Balakrishnan et al. (2019), with the same configuration of hyper-parameters. The experiments with pretrained models implement all above mentioned methods with BART (Lewis et al., 2020). We use the open source `fairseq` implementation (Ott et al., 2019). More specific configuration details of these two models are listed in Appendix A.

3.2 Data Efficiency Study

Figures 1 and 2 show the comparisons among the four training strategies on tree accuracy and BLEU score as a function of the amount of parallel data available. We can clearly see that ST-CD always surpasses the other three self-training methods. Meanwhile, the ST-CD lines are much flatter, indicating better data-efficiency, especially for tree accuracy with less parallel data. In particular, ST-CD achieves a considerable tree accuracy of 90% and 97% with LSTM and BART respectively, using only 1% of the parallel data (253 items). Using 100% of the data, ST-CD sets a new state-of-the-art in tree accuracy and BLEU, exceeding Rao et al.'s (2019) more complex tree-to-sequence method.[3]

Notably, with LSTM vanilla decoding, ST-CD needs only 20% of the parallel data to achieve comparable performance to LBL trained on all the parallel data.[4] More remarkably, BART vanilla decoding ST-CD needs only 2% of the parallel data to achieve essentially comparable performance to LBL trained on all the parallel data.[5] At this data efficiency level, tree accuracy exceeds 97% using just over 500 training samples, while Arun et al.'s (2020) results on the same dataset are under 90% despite using four times as much data. This is a key result since vanilla decoding is so much faster than constrained decoding, and latency is an important consideration for dialogue systems. For example, in our experiments using a single NVIDIA V100, the speed of LSTM vanilla decoding was 925.01 responses/s, or 37,973.22 tokens/s, while the speed of constrained decoding was 12.76 responses/s, or 532.61 tokens/s. This translates to an average of 80ms per response for constrained decoding, which is a barrier to production for systems with a strict latency budget. For BART, the speed of vanilla decoding was 25.17 responses/s, or 1565.75 tokens/s, while the speed of constrained decoding was 1.82 responses/s, or 113.92 tokens/s. As such, BART with vanilla decoding could be suitable in some production settings; alternatively, one could pursue knowledge distillation techniques as in Arun et al. (2020).

Although not as effective as ST-CD, ST-RMR also consistently surpasses ST-VAN and LBL. Moreover, it can also be used in more conventional

[2]Code is available at https://github.com/znculee/TreeNLG and https://github.com/znculee/TreeNLG-BART. See appendix for further details to enhance reproducibility.

[3]Results of using constrained decoding at runtime are shown in Figure 5 and Figure 6 in the appendix.

[4]With 20% of the parallel data, ST-CD exceeds LBL in tree accuracy while trailing it slightly in BLEU.

[5]Confirmed in significance tests on tree accuracy and human evaluation later in this section.

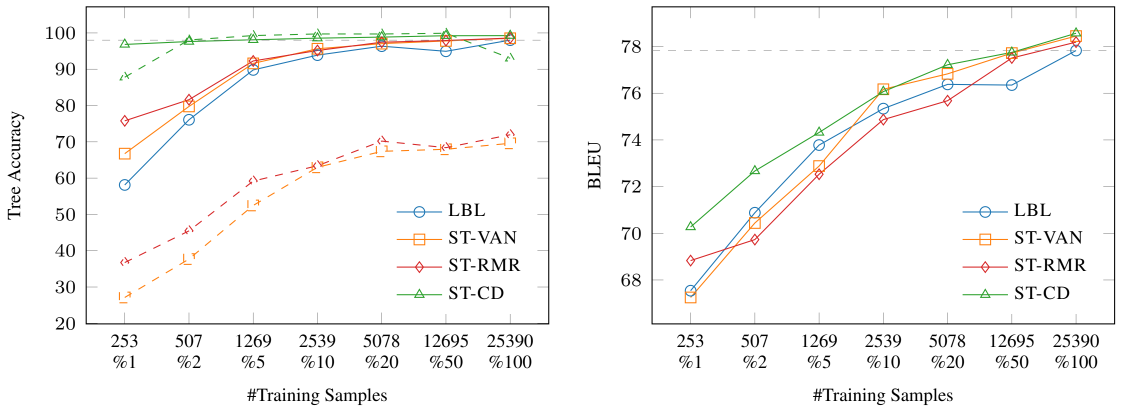

Figure 2: Tree accuracy and BLEU scores of the BART base model and three self-training strategies by parallel training data size with vanilla decoding on the conversational weather dataset. Tree accuracy on pseudo-labeled data is indicated by the same color dashed lines.

settings where the response text in the training data has no semantic annotations, and thus decoding is into plain text. As shown in Figure 3 (appendix), using auto–tree accuracy, ST-RMR can improve data efficiency when constrained decoding cannot be used. Note, however, that decoding into plain text consistently trails in auto–tree accuracy compared to decoding into annotated text.

3.3 How Does Self-Training Help?

Theoretically, self-training should be more helpful when the base model can produce higher quality pseudo-labeled data. As shown in Figures 1 and 2, tree accuracy on pseudo-labeled samples generated by ST-CD is much higher than other self-training methods, which illustrates why it yields much better tree accuracy and BLEU scores on the test set. Also note that the pseudo-labeled tree accuracy is much lower than the test tree accuracy for ST-VAN and ST-RMR. This may be because the unlabeled MRs are created by deletion and thus are somewhat atypical in comparison to the train and test sets.

3.4 Significance Tests

Although the gains in tree accuracy are large with vanilla decoding, to confirm that the gains in Figure 1 and 2 are significant, we have run McNemar's test (McNemar, 1947) comparing ST-CD against LBL as well as ST-VAN. Even when using LSTMs with 100% of the labeled data, the gain in tree accuracy from 94.2% with LBL to 96.6% with ST-CD is highly significant (p=4.30e-15), as is the gain from 95.7% with ST-VAN to 96.6% with ST-CD (p=0.0003). For BART, when using 100% of the labeled data, the gain in tree accuracy from 98.01% with LBL to 99.26% with ST-CD is highly significant (p=1.52e-7), as is the gain from 98.53% with ST-VAN to 99.26% with ST-CD (p=2.94e-4). Naturally, the gains when using less labeled data are also highly significant. Most interestingly, using only 2% of the labeled data with BART ST-CD is not significantly different than using 100% of the labeled data with BART LBL (p=0.68285).

3.5 Expert Evaluation of Correctness

In their experiments, Balakrishnan et al. (2019) found that tree accuracy differences reliably indicated differences in human evaluations of correctness, and in particular that tree accuracy failures nearly always indicated actual correctness errors. To verify these findings in our own targeted expert evaluation, we had two authors (both linguists) judge the correctness of the LSTM and BART models self-trained with constrained decoding using partial parallel data against the supervised baseline using the same partial parallel data and the best supervised model using all the parallel data, where the judges were blind to which model was which. Correctness was judged against the reference text for 50 randomly selected pairs in each condition where the items differed in tree accuracy. For each pair, the judges indicated whether the first item was better than, the same as or worse than the second item. 3-way agreement was 79% for correctness between the judges; moreover, when excluding any 'same' judgments, the judges agreed in all but one case. After the judgments were collected, we calculated how well they agreed with tree accuracy, excluding the indeterminate 'same' judgments. Agreement was quite high, reaching

90% for one judge and 88% for the other. (Further details are in Appendix B.) Given this high level of agreement with the automatic tree accuracy measure along with the highly significant differences in tree accuracy, we focused our human evaluation on investigating whether the observed differences in BLEU scores indicated important differences in grammaticality.

3.6 Human Evaluation of Grammaticality

While the BART ST-CD-02 and LSTM ST-CD-20 models achieved comparable or better levels of tree accuracy in comparison to their LBL-100 (full-data) counterparts, they trailed somewhat in BLEU scores. Looking at the outputs of the self-trained models with the worst BLEU scores, we found that the responses were mostly good, only suffering from clear grammaticality issues infrequently. To confirm these observations, we conducted a human evaluation using the responses generated by the BART ST-CD-02 and LSTM ST-CD-20 models on 333 randomly selected test items, along with the responses for the same items for the best and worst supervised models by BLEU score, namely BART LBL-100 and LSTM LBL-01. Mixed in with the responses of each model were 75 check items, 25 of which were grammatical and 50 of which we intentionally made ungrammatical.

Using these samples, we ran an experiment on Amazon Mechanical Turk involving 16 unique participants. The participants in the experiment were pre-filtered by selecting those with an approval rate of at least 95%. Each participant was shown our grammaticality guidelines, which were based on Arun et al.'s (2020) and available for review at all times during the experiment. They were subsequently asked to take a quiz. Those who scored 80% or more on the quiz were selected for further participation. To encourage careful engagement with the task, we offered bonus payments to those who performed well on the check items. The experiments were carried out with Institutional Review Board approval, and all participants were paid above minimum wage for our locale.

Agreement with the check items was quite robust, with all participants well above chance, though there were some outliers with respect to check item agreement. This indicates that the judgments were somewhat noisy. Each item received 3 judgments, and the items were assigned the majority judgment for analysis purposes. Judgments of ungrammaticality were accompanied by brief reasons; discrepancies between judgments primarily reflected difficulty in applying the guidelines regarding punctuation.

Our results indicate that 4.8% of the BART ST-CD-02 items were judged ungrammatical, not far from the error rates of 3.9% for LSTM ST-CD-20 and 3.0% for BART LBL-100. By contrast, 11.4% of the LSTM LBL-01 items were judged ungrammatical. Pairwise comparisons using McNemar's test only revealed statistically significant differences for the LSTM LBL-01 model: it was judged significantly worse than the 3 other models ($p < 0.003$ in all cases), while none of the other systems were significantly different ($p > 0.3$ in all cases).

3.7 Qualitative Analysis

The most frequent grammaticality issue, especially for LSTM ST-CD-20, was missing punctuation between independent clauses, as shown in (a) in Table 2. Other errors included occasional agreement errors or missing constituents, as in (b). Example correctness errors appear in Table 3; they usually involved missing information, but sometimes also repeated or extra information.

3.8 E2E Confirmation

We also evaluate our strategies on the enhanced E2E dataset. As shown in Figure 4 in Appendix, we can draw the same general conclusions regarding data efficiency as with the conversational weather dataset.[6] Both constrained decoding and reverse model reranking improve upon vanilla self-training, with constrained decoding being more effective when using less parallel data. Notably, for LSTM models, with vanilla decoding at runtime, tree accuracy and BLEU of using self-training with constrained decoding and 20% of the parallel data (ST-CD-20) are essentially identical to the supervised model using all the available data (LBL-100). For BART models, the performance of ST-CD-02 is also very similar to the one of LBL-100: While the BLEU score of ST-CD-02 is slightly lower than that of LBL-100, it is still very high, and the tree accuracy of ST-CD-02 is slightly higher than the tree accuracy of LBL-100.

[6]Note that the BLEU scores here are calculated in the same generous way as in Balakrishnan et al.'s (2019) evaluation. In particular, since multiple test MRs in the enhanced data have the same original MR, we select the best generation of the same original MR using NLTK's (Bird et al., 2009) implementation of sentence BLEU on multi-references.

Index	System	Error	Reason
(a)	LSTM ST-CD-20	No , the forecast does not call for sunny skies expect partly cloudy skies	Punctuation is missing before *expect*.
(b)	BART ST-CD-02	Today in ARG_CITY will have a high of ARG_TEMP_HIGH and a low of ARG_TEMP_LOW	Missing subject.

Table 2: Examples of grammaticality errors

Index	System	Error	Reference
(a)	LSTM LBL-20	Yes , it will be mostly sunny today in your area	Yes , it will be mostly sunny today and ARG_WEEKDAY in your area
(b)	LSTM LBL-100	Yes , light rain is likely today , and light thunderstorms and rain are likely on ARG_WEEKDAY and light thunderstorms and rain are likely on ARG_WEEKDAY	Yes , light rain is likely today . ARG_WEEKDAY will also have light rain and light thunderstorms and rain are likely on ARG_WEEKDAY

Table 3: Examples of correctness errors

4 Related Work

There is a much more established tradition of using self-training in parsing, where McClosky et al. (2006) and subsequently others have shown that that self-training can yield substantially improved parsing accuracy. In NLG, Kedzie and McKeown (2019) and Qader et al. (2019) pursue self-training for data efficiency but only using flat input representations and without constrained decoding, as noted earlier. Qader et al. (2019) develop a sophisticated, joint method of self-training NLG and NLU models. Kedzie and McKeown (2019) make use of noise injection sampling and NLU models to create new MR-text pairs, where the new MRs often contain fewer slots than the original MR; here, we similarly create new, simpler MRs, but do so directly by just deleting nodes in the input trees. Likewise, our general approach to self-training (He et al., 2020) is much simpler than in Chang et al.'s (2021) work, where they generate new text samples using GPT-2 (unconditioned on any input) then pair them with data samples. Earlier, Chisholm et al. (2017) train NLG and NLU models that share parameters to reduce the risk of hallucination. Our iterative method of training forward and reverse seq2seq models instead draws from Yee et al.'s (2019) reverse model reranking approach and is much simpler to implement. Additionally, Nie et al. (2019) apply self-training to a NLU model to reduce the noise in the original MR-text pairs in order to reduce the hallucination problem in NLG, but they do not investigate data efficiency issues. Also related is work on back-translation (Sennrich et al., 2016) in MT, which starts from the assumption that there is much target side data; by contrast, self-training assumes there is much source side data, as is the case with our task (where new unlabeled MRs can be easily created).

More recent work takes advantage of pre-trained language models to develop few-shot NLG methods. Chen et al. (2019) show impressive results with just 200 training items using a specialized table encoder with GPT-2, while Peng et al. (2020) use cross-domain training (an orthogonal approach) together with GPT-2; neither investigates more challenging compositional inputs. Although Arun et al. (2020) do use BART on compositional inputs, their tree accuracy levels are much lower even when using considerably more data.

More generally, reverse (or reconstructor) models have taken on greater theoretical interest thanks to Rational Speech Act (RSA) theory (Frank et al., 2009) and have recently proved useful in NLG (Fried et al., 2018; Shen et al., 2019). Our approach differs in using reverse models during self-training rather than at runtime. Work on combining parsing and generation for ambiguity avoidance goes back much farther (Neumann and van Noord, 1992), with managing the trade-off between fluency and ambiguity avoidance a more recent topic (Duan and White, 2014) that we also leave for future work. Constrained decoding (Balakrishnan et al., 2019) is inspired by coverage tracking in grammar-based approaches to realization (Kay, 1996; Carroll and Oepen, 2005; White, 2006); its use during self-training is novel to this work.

5 Conclusion and Future Work

In this paper, we have shown that using self-training with constrained decoding in compositional neural NLG can deliver large gains in data efficiency, enabling seq2seq models to achieve satisfactory quality using vanilla decoding with much less annotated data. The idea of using constrained decoding with self-training rather than for runtime inference is a very simple one, but ours is the first paper to investigate the idea, and we show via thorough experimentation and evaluation that it works remarkably well. In our experiments, we found that LSTM models trained from scratch can increase data efficiency by a factor of at least 5, while pretrained BART models yielded a 50 times increase, achieving essentially comparable levels of correctness and grammaticality using only 2% of the existing training data. As such, the approach promises to help pave the way towards developing systems with mere hundreds rather than tens of thousands of annotated samples, potentially eliminating the need for crowdsourcing in system development. In future work, it would be exploring ways of at least partially automatically adding semantic annotations to the target texts using methods that treat such annotations as latent (Shen et al., 2020; Xu et al., 2021) to facilitate using our approach on a new task or dataset.

Acknowledgements

We thank that the Ohio Super Computer Center (Center, 1987) supports us sufficient computational devices for training many large models in our experiments. This research was supported by a collaborative open science research agreement between Facebook and The Ohio State University. The last author is a paid consultant for Facebook.

References

Ankit Arun, Soumya Batra, Vikas Bhardwaj, Ashwini Challa, Pinar Donmez, Peyman Heidari, Hakan Inan, Shashank Jain, Anuj Kumar, Shawn Mei, Karthik Mohan, and Michael White. 2020. Best practices for data-efficient modeling in NLG:how to train production-ready neural models with less data. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 64–77, Online. International Committee on Computational Linguistics.

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.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Giuseppe Carenini and Johanna D. Moore. 2006. Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170:925–952.

John Carroll and Stefan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proc. IJCNLP-05*.

Ohio Supercomputer Center. 1987. Ohio supercomputer center.

Ernie Chang, Xiaoyu Shen, Dawei Zhu, Vera Demberg, and Hui Su. 2021. Neural data-to-text generation with LM-based text augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 758–768, Online. Association for Computational Linguistics.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2019. Few-shot nlg with pre-trained language model.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from Wikidata. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 633–642, Valencia, Spain. Association for Computational Linguistics.

Vera Demberg, Andi Winterboer, and Johanna D Moore. 2011. A strategy for information presentation in spoken dialog systems. *Computational Linguistics*, 37(3):489–539.

Manjuan Duan and Michael White. 2014. That's not what I meant! Using parsers to avoid structural ambiguities in generated text. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 413–423, Baltimore, Maryland. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurcicek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the E2E NLG Challenge. In *Proc. of the 11th International Conference on Natural Language Generation*, pages 322–328, Tilburg, The Netherlands. Association for Computational Linguistics.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2019. Evaluating the state-of-the-art of end-to-end natural language generation: The E2E NLG Challenge. *arXiv preprint arXiv:1901.11528*.

Michael Frank, Noah Goodman, Peter Lai, and Joshua Tenenbaum. 2009. Informative communication in word production and word learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, pages 1228–1233.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018. Unified pragmatic models for generating and following instructions. 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 1951–1963, New Orleans, Louisiana. 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*.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 200–204, Santa Cruz, California, USA. Association for Computational Linguistics.

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*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training 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.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Hongyuan Mei, Mohit Bansal, and R. Matthew Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730. Association for Computational Linguistics.

Gunter Neumann and Gertjan van Noord. 1992. Self-monitoring with reversible grammars. In *COLING 1992 Volume 2: The 15th International Conference on Computational Linguistics*, pages 700–706.

Feng Nie, Jin-Ge Yao, Jinpeng Wang, Rong Pan, and Chin-Yew Lin. 2019. A simple recipe towards reducing hallucination in neural surface realisation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2673–2679, Florence, Italy. Association for Computational Linguistics.

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*.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot natural language generation for task-oriented dialog.

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.

Jinfeng Rao, Kartikeya Upasani, Anusha Balakrishnan, Michael White, Anuj Kumar, and Rajen Subba. 2019. A tree-to-sequence model for neural NLG in task-oriented dialog. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 95–100, Tokyo, Japan. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative 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 4060–4067, Minneapolis, Minnesota. Association for Computational Linguistics.

Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. 2020. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165, Online. Association for Computational Linguistics.

Marilyn Walker, Amanda Stent, Francois Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research (JAIR)*, 30:413–456.

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

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

Michael White, Robert A. J. Clark, and Johanna D. Moore. 2010. Generating tailored, comparative descriptions with contextually appropriate intonation. *Computational Linguistics*, 36(2):159–201.

Xinnuo Xu, Ondřej Dušek, Verena Rieser, and Ioannis Konstas. 2021. AggGen: Ordering and aggregating while generating. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1419–1434, Online. Association for Computational Linguistics.

Kyra Yee, Yann Dauphin, and Michael Auli. 2019. Simple and effective noisy channel modeling for neural machine translation. 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 5696–5701, Hong Kong, China. Association for Computational Linguistics.

A Reproducibility Details

For LSTM models, the word embedding and hidden size dimensions are 300 and 128 respectively, and the decoder output embedding size is 512. The dropout rate for both encoder and decoder is 0.2. There are no more than 128 sentences in a batch. Training uses early stopping when the validation loss has not improved for the last 20 epochs. The learning rate is 0.001, and the scheduler is *ReduceLROnPlateau* whose factor is 0.1 and patience is 3. The maximum output length is 2 times source length plus 50, and the beam size is 5. The loss function is optimized with Adam (Kingma and Ba, 2014), where $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

For BART models, we use the BART-Large model available in the fairseq, which 12 encoder and decoder layers.[7] The dropout rate for both encoder and decoder is 0.1. There are no more than 2048 tokens in a batch. Training uses early stopping when the validation loss has not improved for the last 20 epochs. The learning rate is 0.00003, and the scheduler is *polynomial decay* with 1000 warm updates. The maximum output length is 1024. The loss function is optimized with Adam (Kingma and Ba, 2014), where $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

For every experiment, the computing infrastructure we used is an NVIDIA V100 GPU and an Intel(R) Xeon(R) Platinum 8268 CPU @ 2.90GHz CPU. The numbers of trainable parameters of LSTM models for weather and E2E datasets are 2,212,928 and 3,079,256 respectively. Training a LSTM model on the full weather dataset takes around 0.5k seconds for 38 epochs. Training a LSTM model on the pseudo-labeled weather dataset takes around 3.4k seconds for 57 epochs. Training and validation loss at convergence is around 1.8. The speed of vanilla decoding was 37,973 tokens/s, and the speed of constrained decoding was 532.61 tokens/s. The numbers of trainable parameters of BART models for weather and E2E datasets are both 406,290,432. Training a BART model on the full weather dataset takes around 10k seconds for 21 epochs. Training a BART model on the pseudo-labeled weather dataset takes around 42k seconds for 20 epochs. Training and validation loss at convergence is around 2.1. The speed of vanilla decoding was 25.17 responses/s, or 1565.75 tokens/s, and the speed of constrained decoding was 1.82 responses/s, or 113.92 tokens/s.

The source code and data for reproducing all experiments in this paper are submitted in the supplementary materials and will

[7] https://github.com/pytorch/fairseq/tree/master/examples/bart

be released upon acceptance. The dependencies are specified in `requirements.txt`. Code usage instructions are in `README.md` and `self-training/README.md`.

B Details on Expert Evaluation of Correctness

Table 4 shows the detailed breakdown of agreement between the expert judges and tree accuracy. We can observe that agreement with tree accuracy is higher with LSTM models than with BART, and higher where there is a significant difference in tree accuracy than in the one case where there was no significant difference (BART ST-CD-02 vs. BART LBL-100). For this comparison, there were relatively few discrepancies in tree accuracy to sample from, and the items in question likely represent somewhat unusual cases. In examining the handful of cases where the judges agreed but did not agree with tree accuracy, about half were real errors where BART's words did not match the non-terminals (influenced by its pre-trained knowledge), while the other half had (presumably rare) errors in the input or reference. It is not surprising that tree accuracy would be somewhat less reliable with BART, as it relies on its pre-trained knowledge as well as the input in making generation choices. For example, in one case the BART ST-CD-02 model output, "It's not expected to be warm tomorrow morning in your area. The temperature will drop to __ARG_TEMP__ tomorrow." Here, it seems that BART inferred that if it won't be warm tomorrow, that may well be because the temperature is dropping. However, "will drop" is not part of the input and may or may not be factual. Since these words appear outside of the non-terminal signaling the low temperature in the output, they are not checked by tree accuracy, and thus this error is missed.

LSTM	ST-CD-20 vs.	
	LBL-20	LBL-100
Judge-1	36/39	26/29
Judge-2	40/40	25/27

BART	ST-CD-02 vs.	
	LBL-02	LBL-100
Judge-1	36/37	21/31
Judge-2	36/37	18/29

Table 4: Agreement rate of human evaluation of correctness with tree accuracy (excluding indeterminate 'same' judgments)

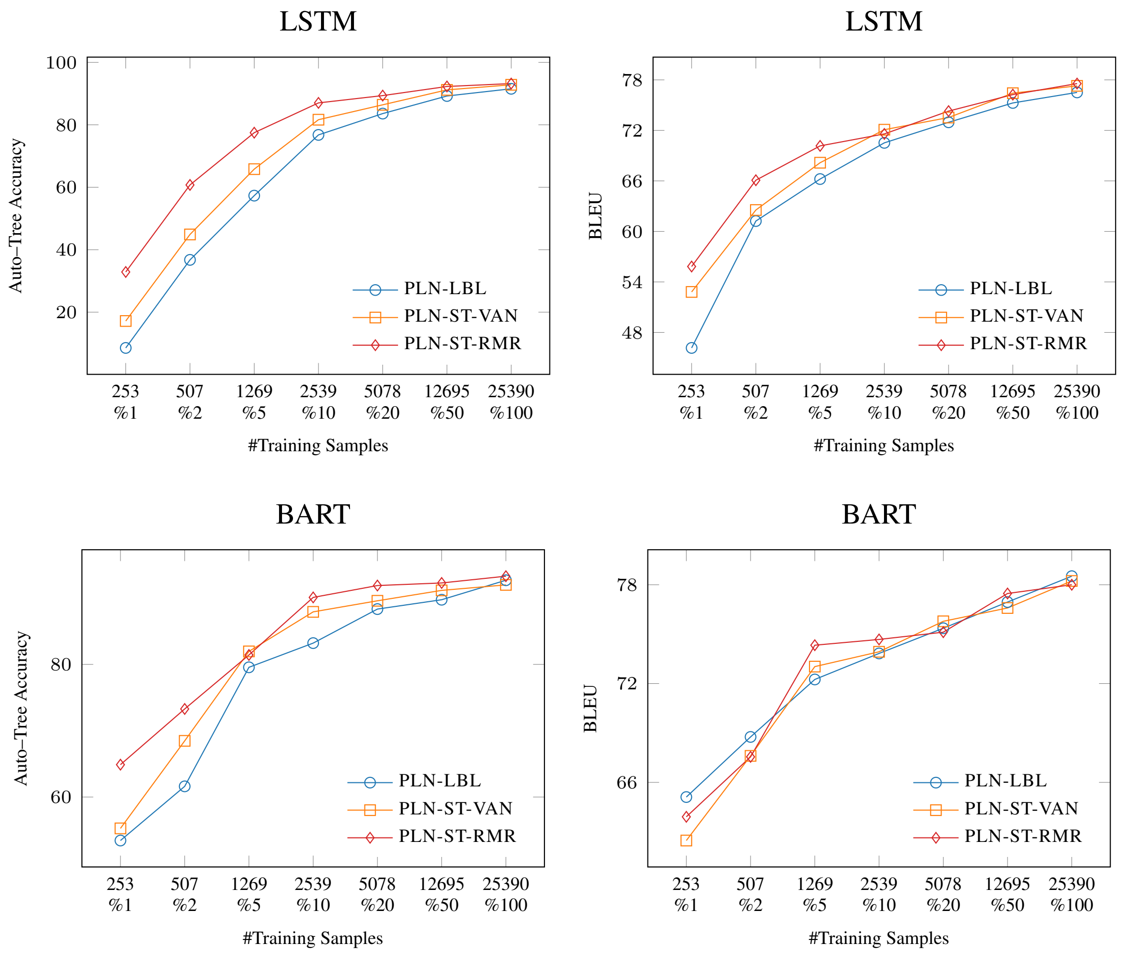

Figure 3: Auto–tree accuracy and BLEU scores of LSTM and BART models and two self-training methods by parallel training data size with vanilla decoding of plain (PLN) text on the conversational weather dataset. For comparison, auto–tree accuracy of LSTM and BART on the test set references are 92.85 and 93.18 respectively.

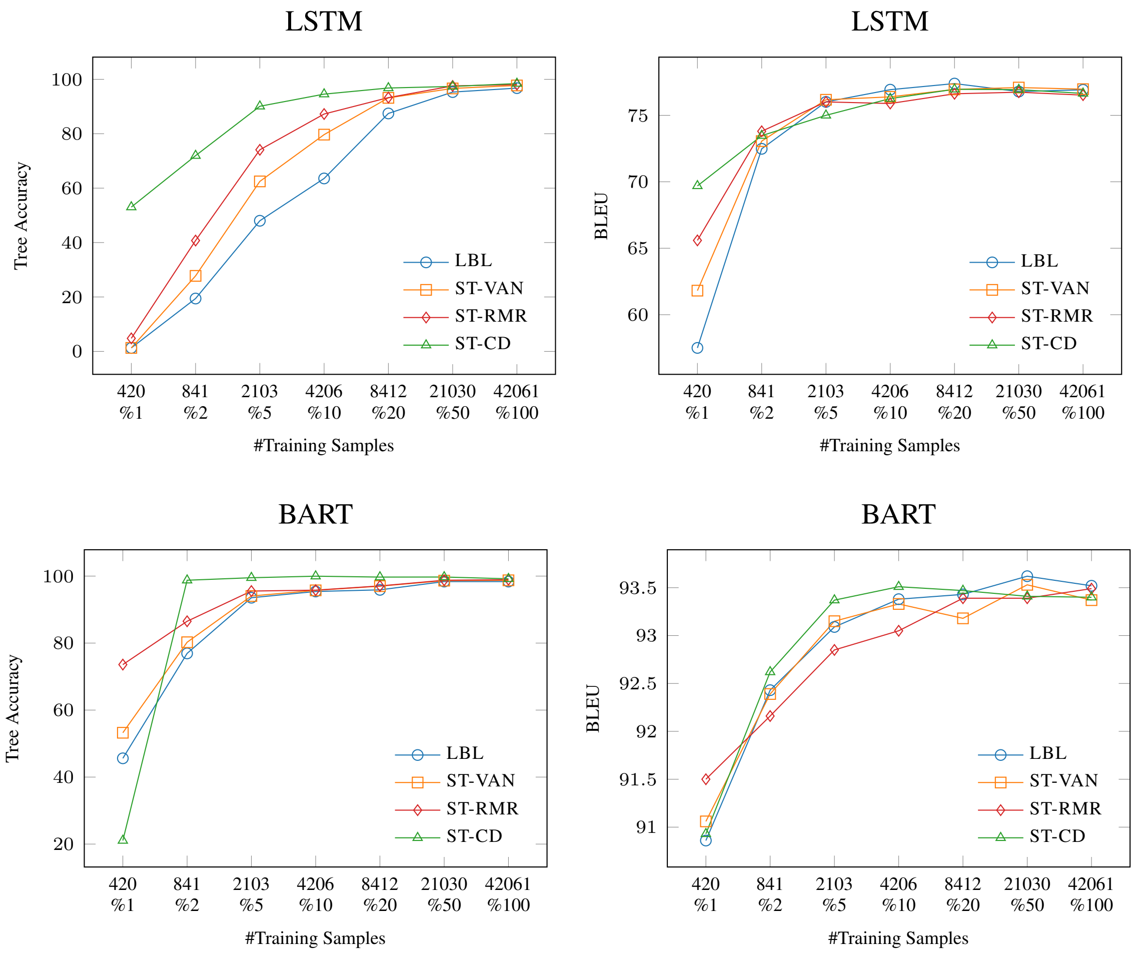

Figure 4: Tree accuracy and BLEU scores of LSTM and BART models and two self-training strategies by parallel training data size with vanilla decoding on the enhanced E2E dataset. The self-training results here are measured on the first iteration.

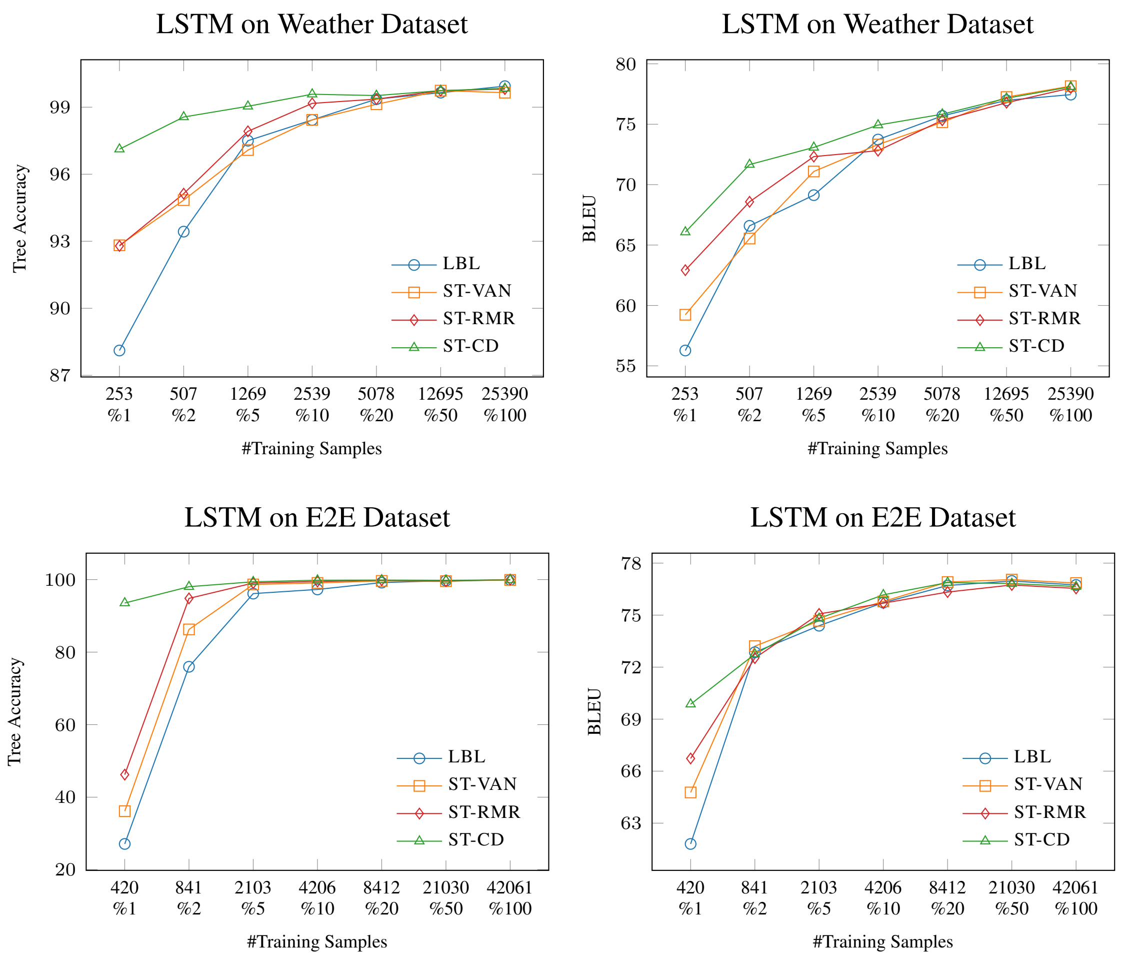

Figure 5: Tree accuracy and BLEU scores of LSTM and two self-training strategies by parallel training data size with **constrained decoding** at runtime on the conversational weather dataset and the enhanced E2E dataset. The self-training results of the enhanced E2E dataset are measured on the first iteration.

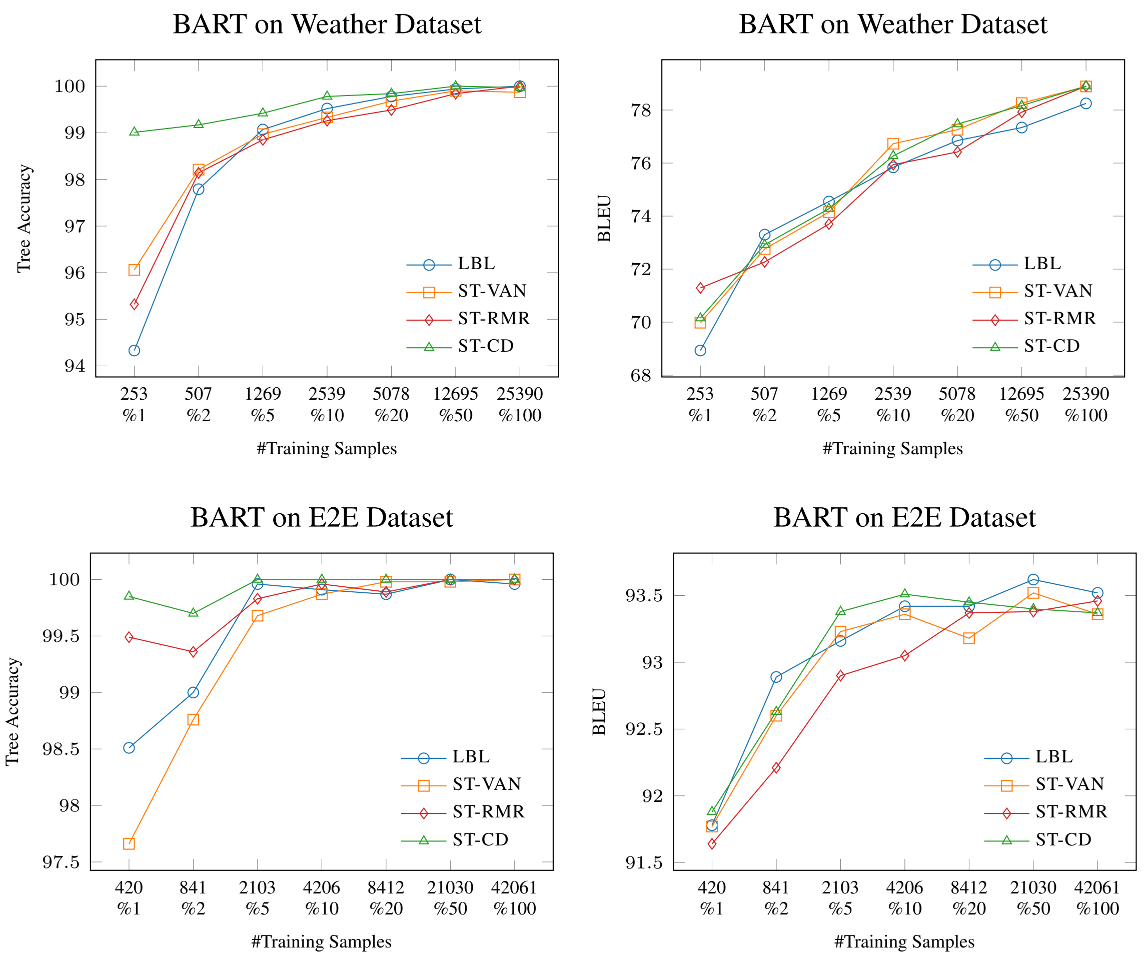

Figure 6: Tree accuracy and BLEU scores of BART and two self-training strategies by parallel training data size with **constrained decoding** at runtime on the conversational weather dataset and the enhanced E2E dataset. The self-training results of the enhanced E2E dataset are measured on the first iteration.