Hierarchical Recurrent Aggregative Generation for Few-Shot NLG

Anonymous ACL submission

Abstract

Large pretrained models enable transfer learning to low-resource domains for language generation tasks. However, previous end-to-end approaches do not account for the fact that some generation sub-tasks, specifically aggregation and lexicalisation, can benefit from transfer learning to different extents. To exploit these varying potentials for transfer learning, we propose a new hierarchical approach for few-shot and zero-shot generation. Our approach consists of a three-moduled jointly trained architecture: the first module independently lexicalises the distinct units of information in the input as sentence sub-units (e.g. phrases), the second module recurrently aggregates these sub-units to generate a unified intermediate output, while the third module subsequently post-edits it to generate a coherent and fluent final text. We perform extensive empirical analysis and ablation studies on few-shot and zero-shot settings across 4 datasets. Automatic and human evaluation shows that the proposed hierarchical approach is consistently capable of achieving state-of-the-art results when compared to previous work.[1]

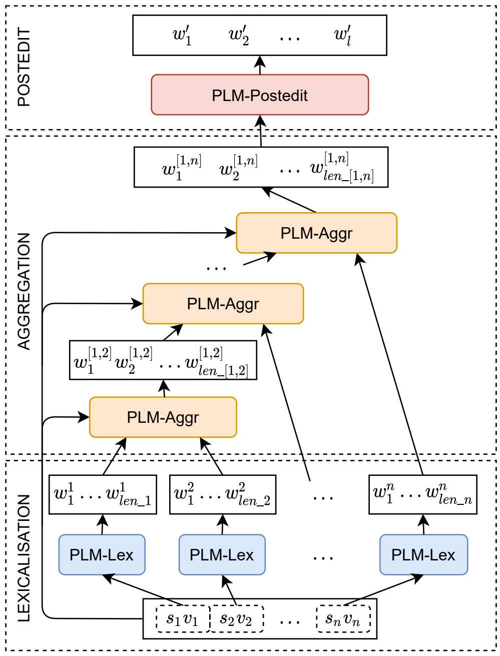

Figure 1: Structure of Hierarchical Recurrent Aggregative Generation (HRAG). The lexicalisation PLM generates one sub-phrase per attribute-value pair. The aggregation PLM recurrently combines sub-phrases and the post-edit PLM rephrases them into a fluent output.

1 Introduction

The recent development of large pretrained language models (PLMs; i.e. BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020), T5 (Raffel et al., 2020)) has caused a shift of interest in the research community towards domain adaptation and transfer learning. For the task of concept-to-text natural language generation (NLG), wherein the aim is to generate a natural language text that describes the semantic content of an abstract structured machine-readable input (Meaning Representation; MR), transfer learning from PLMs has become a popular and high performing approach with 13 out of the 15 participating teams in the latest WebNLG+ Shared Task (Ferreira et al., 2020) employing a fine-tuned pretrained model as their main submitted system. Specifically, T5-based systems achieved a human evaluation ranking on par with the ground truth in terms of fluency and adequacy. Transfer learning from PLMs also enables training on few-shot and zero-shot settings, i.e. when sufficient in-domain data are unavailable. Prominent and relevant examples include machine translation (Zoph et al., 2016; Brown et al., 2020) and NLG for task-oriented dialogues (Peng et al., 2020).

This paper focuses on concept-to-text NLG, where recent machine learning and in extension transfer learning approaches adopt an end-to-end architecture (Peng et al., 2020) that inputs the full meaning representation and produces the full out-

[1] Code will be made public on acceptance of the paper.

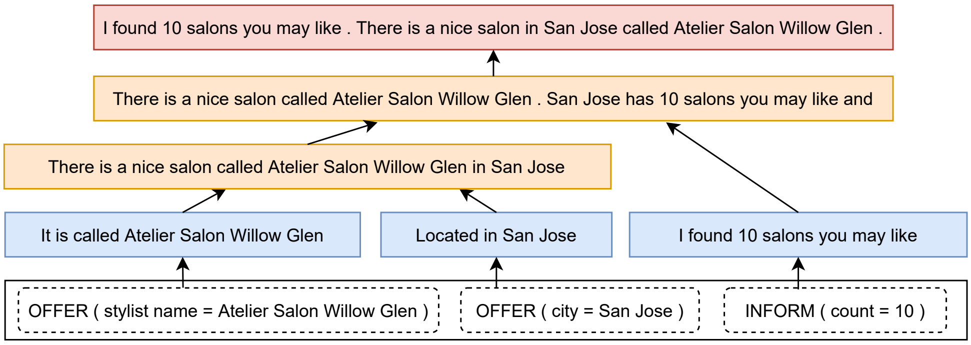

Figure 2: Example of lexicalisation (in blue), recurrent aggregation (in orange), and post-editing (in red) stages.

put text. In such end-to-end models, the traditional sub-tasks (Reiter and Dale, 2000) involved in language generation (i.e. planning, lexicalisation, aggregation, referring expression generation, and surface realisation) are performed implicitly. However, we posit that some of these sub-tasks, specifically lexicalisation (i.e. choice of vocabulary) and aggregation (i.e. process of combining simpler sentence structures to form complex ones), exhibit varying potential for exploiting transfer learning as the former is more domain-specific than the latter. For example, it is more difficult to exploit transfer learning for lexicalisation since if certain words are not already associated with a particular MR input, few-shot learning may not be able to create a strong association through the limited data. This is further exacerbated in zero-shot learning. On the other hand, the knowledge required to form complicated sentence structures and apply aggregation strategies is more commonly shared between domains and would benefit more from transfer learning.

We aim to exploit these differing potentials for transfer learning in few-shot and zero-shot generation, via a new hierarchical approach to concept-to-text NLG. Specifically, we propose Hierarchical Recurrent Aggregative Generation (HRAG), a three-moduled architecture where the first module is in charge of independently lexicalising each unit of information in the input as a sub-phrase (e.g. a phrase expressing that unit of information alone), the second module is responsible for recurrently aggregating these sub-units to generate a unified text, and the third module rephrases it to produce a coherent and fluent output; see Figure 1. These are jointly trained via a loss that combines their discrete objectives. Concept-to-text is ideal for HRAG as MRs can be split into attribute-value pairs that vaguely correspond to output sub-phrases.

In this paper, we (i) present Hierarchical Recurrent Aggregative Generation and experimentally demonstrate the benefits of separately applying transfer learning to language generation sub-tasks; (ii) facilitate the model's training by inferring module-specific training signal from the available output targets; (iii) provide extensive empirical analysis and ablation studies on few-shot and zero-shot settings across 4 datasets, one of which we adapt ourselves for few-shot learning; (iv) perform human evaluation comparing our proposed approach to previous work on few-shot generation. Our automatic and human evaluation results show that our hierarchical approach achieves state-of-the-art results when compared against previous work.

2 Method

Figure 1 shows the overall structure of the proposed hierarchical model HRAG. Its three modules are in charge of lexicalisation, aggregation and post-edit, and are inspired by traditional NLG stages and their specific potential for transfer learning in a few-shot setting. Figure 2 shows an example of how the outputs of each stage are formed.

2.1 Input segmentation

In a pre-processing step, the input MR is divided into individual attribute value pairs $s_x v_x$ each corresponding to one distinct *fact* (i.e. unit of information). Concept-to-text generation is particularly fitted to our approach as the input MR is usually straightforwardly divisible into distinct facts. To elaborate, a typical input MR consists of one or more predicates that denote the communicative goal of the sentence, followed by a set of attribute-value pairs that correspond to the information that should be expressed in the final text.

For example, in Figure 2 the input MR describes that the text should offer/suggest to the user a stylist named "Atelier Salon Willow Glen" that is in the city of "San Jose', and also inform them that it has found "10" salons that match their criteria. We assume that each attribute-value pair corresponds to one distinct fact which is expressed as a sub-phrase of the final output, e.g. CITY = SAN JOSE loosely corresponds to the sub-phrase "in San Jose".

2.2 Lexicalisation

The next stage is lexicalisation, i.e. the process of selecting the required vocabulary to express the input. HRAG's respective module achieves this by independently generating a corresponding phrase $w_1^x \dots w_{len\_x}^x$ for each input fact $s_x v_x$, e.g. "located in San Jose" should be generated from input CITY = SAN JOSE in Figure 2. We opt to generate from single facts, disconnected from their MR context, as it makes it easier for the model to associate them with their relevant vocabulary. This might lead to the loss of informative context, but HRAG reintroduces context in a later stage. Additionally, having a single fact input facilitates transfer learning in the few-shot setting since any previous context may be irrelevant to new domains. A final benefit is that such input is more robust to unseen facts, as any unknown attributes will only affect the corresponding sub-phrase and will not interfere with the generation from other facts.

In contrast, due to considering the whole input at once, previous end-to-end models need to be exposed to a lot of different combinations and orderings of attribute-value slots, to sufficiently associate complex input MRs with the output text. In few-shot settings, this becomes an issue as available MR combinations during training are limited.

2.3 Recurrent aggregation

In this stage, the generated sub-phrases of the lexicalisation module are ordered based on the input's original order, and input into the aggregation layer one at a time in a recurrent fashion. At the first step, the first two sub-phrases $w_1^1 \dots w_{len\_1}^1$ and $w_1^2 \dots w_{len\_2}^2$, and the correspondent attribute-value pairs $s_1v_1\ s_2v_2$, are input into the aggregation layer to produce the combined sub-phrase $w_1^{[1,2]} \dots w_{len\_[1,2]}^{[1,2]}$ (see Figure 1). For example, the sub-phrases "it is called Atelier Salon Willow Glen" and "located in San Jose" are combined to form "there is a nice salon called Atelier Salon Willow Glen located in San Jose" as shown in Figure 2.

At each subsequent step r the input of the aggregation module consists of the concatenation of the previously aggregated sub-phrases $w_1^{[1,r-1]} \dots w_{len\_[1,r-1]}^{[1,r-1]}$, the current sub-phrase $w_1^r \dots w_{len\_r}^r$, and the correspondent attribute-value pairs $s_1v_1\ s_2v_2 \dots\ s_rv_r$, to produce the combined sub-phrase $w_1^{[1,r-1]} \dots w_{len\_[1,r]}^{[1,r]}$. The aggregation module is called recurrently until all the sub-phrases generated by the lexicalisation module are combined into a single output $w_1^{[1,n]} \dots w_{len\_[1,n]}^{[1,n]}$.

Each distinct aggregation layer has the advantage of being able to disassociate (to some extent) from the specific semantics of the input and direct its attention on how to combine (and copy over) the sub-phrases of the lexicalisation module. This is further enhanced by the recurrent structure of the proposed aggregation layer which permits the model to focus on a limited amount of operations at a time, converging into a final unified output.

2.4 Post-editing

The aggregation layer models are trained to combine sub-phrases into larger sub-phrases and do not necessarily produce a fluent and coherent text complete with appropriate punctuation and devoid of errors. In order to rewrite the aggregated sub-phrases, fix any errors and finalise the text, the post-edit module takes the fully aggregated sub-phrases $w_1^{[1,n]} \dots w_{len\_[1,n]}^{[1,n]}$ and produces the output $w'_1 \dots w'_l$, as seen in the top stage of Figure 2. Being largely domain-agnostic, aggregation and post-edit benefit the most from transfer learning.

2.5 Training, reranking and selection

Each module is built on top of a PLM; these PLMs have separate shared weights per stage and are specifically fine-tuned for that stage. For training, the modules' losses are combined as in Eq. 1:

$$Loss = \frac{1}{n}\sum_n Loss_{lex} + \frac{1}{n-1}\sum_{n-1} Loss_{aggr} + Loss_{pe} \tag{1}$$

where *cross entropy* is used for $Loss_{lex}$, $Loss_{aggr}$ and $Loss_{pe}$, and n the number of units in the MR.

To mitigate any data sparsity issues, we employ language agnostic delexicalisation (Zhou and Lampouras, 2021) for the lexicalisation and aggregation modules, with relexicalisation performed before post-edit. Briefly, any input value that is determined to occur in the text (via embedding similarity) is delexicalized. In addition, to minimise the error propagated between layers, each module generates multiple hypotheses per input and forward the hypothesis with the least slot error rate to

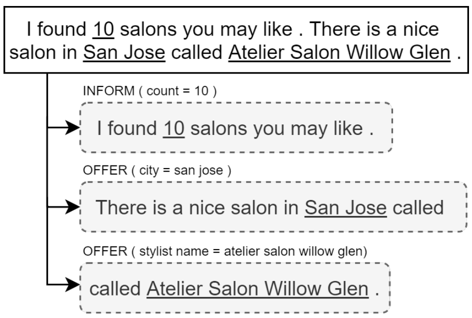

Figure 3: Example of sub-phrase target inference for training the lexicalisation module. The underlined values are matched with the input values.

the next iteration/module, where slot error rate is defined as the percentage of values in the input that are missing, repeated or hallucinated in the output.

2.6 Inferring labels

Ideally, the PLMs that are used in HRAG's different modules would be fine-tuned on stage-specific parallel input and target data. However, while the post-edit module can be trained against the dataset's final output target, such direct annotations for the first two modules are not readily available. To overcome this, we adopt a distant supervision approach to automatically extract stage-appropriate training signals from the existing data.

For the lexicalization stage, we extract sub-phrase targets from the output target that weakly correspond to the individual facts; this process is depicted in Figure 3. Given an MR, we first determine occurrences of its values in the output target via language agnostic delexicalisation. If the value is not matched, we repeat the process using the attribute instead; this is useful for some boolean attributes (e.g. "accepts credit cards = yes"). If a match is still not found, we assume that the fact is not present in the output target, and we ignore that attribute-value pair from the input during training.

For each fact $s_x v_x$, the corresponding target sub-phrase is set to include the matched value of v_x and all words preceding and following it until either a punctuation mark or another matched value is reached. This will cause some overlap between the inferred sub-phrase targets but ensures that all the relevant vocabulary is included in each fact's target. While using this noisy training signal may encourage some hallucinations of irrelevant input, in preliminary experiments this strategy worked better than alternatives; the aggregation layer proved robust enough to ignore irrelevant or repeated words that were output from the lexicalisation layer.

Using the aforementioned value matching, we can similarly infer targets for the aggregation layers. However, to facilitate the process, the order in which lexicalisation sub-phrases are aggregated (see Section 2.3) needs to be fixed to the appearance order of the corresponding matched values in the output target. Given the example of Figure 3, the order would be INFORM (COUNT = 10) > OFFER (CITY = SAN JOSE) > OFFER (STYLIST NAME = ATELIER SALON WILLOW GLEN).

The aggregation targets are then inferred as such: for every aggregation group $s_1 v_1$ $s_2 v_2$ ... $s_r v_r$, the target consists of a subphrase of the output target, from its beginning, including the words of the last matched value v_r, and until either a punctuation or another matched value is reached after that point. Again following the example of Figure 3, the aggregation target for INFORM (COUNT = 10) + OFFER (CITY = SAN JOSE) will be *"I found 10 salons you may like. There is a nice salon in San Jose called"*.

We note that this order of lexicalisation sub-phrases is only imposed during training since we are limited by the output target. During testing, as we mentioned in Section 2.5, the generated sub-phrases of the lexicalisation module follow the original input's order. This results in an important discrepancy between the order of sub-phrases that HARG is exposed to during training and inference, which we leave for future work.

3 Experimental Setup

3.1 Datasets

We perform experiments on four datasets: Schema-Guided Dialogue (Rastogi et al., 2020, SGD) with the few-shot splits provided by (Kale and Rastogi, 2020, FewShotSGD), MultiWoZ 2.2 (Zang et al., 2020), FewShotWoZ (Peng et al., 2020) and WebNLG 3.0 (Ferreira et al., 2020). The first three are task-oriented dialogue datasets, that have been adapted to different extents for few-shot learning by previous work. For our experiments, dialogue MRs are linearised as lists of "INTENT (ATTRIBUTE = VALUE)", similar to what is depicted in Figure 2, while utterances are tokenised and lower-cased.

In contrast to the other datasets, WebNLG 3.0 (Ferreira et al., 2020) does not contain dialogues but describes entities from a variety of domains, and consists of sets of RDF triples and corresponding texts in English and Russian; here we use only the English portion. The dataset is organised in subsets based on the number of RDF triples in the input, ranging from 1 to 7. To create appropriate splits for few-shot learning, for each length-specific subset, we identified all unique combinations of RDF properties in the input and limited the dataset to a single (where available) instance per combination. In other words, we kept only 1 instance per property for the 1-triple subset, 1 instance per pair of properties for the 2-triple subset, and so forth. Our splits essentially constitute a 1-shot learning dataset, which we will refer to as FewShotWeb dataset. More details regarding the FewShotWeb splits can be found in appendix A. Preprocessing of the RDF triples and target text were performed similarly to Zhou and Lampouras (2021).

3.2 Automatic metrics

Following related work, to estimate the fluency of the output, we provide results for BLEU-4 (computed with SacreBLEU) (Papineni et al., 2002; Post, 2018), and BLEURT (Sellam et al., 2020) (specifically the *bleurt-base-128* version). We calculate BLEU score over multiple references to mitigate the unreliability of single reference evaluation.

To estimate adequacy, we use Missing Slot Error (MER), computed as the macro-averaged percentage of values in the MR that are missing (i.e. do not appear verbatim) from the output utterance. We should note that MER is an imperfect approximation compared to slot error rate, as it does not account for hallucinations, boolean or no-value attributes. These types of slot errors are difficult to detect in non-delexicalized output, which all systems in our experiments produce. Evaluation is performed consistently across all datasets.[2]

3.3 Systems

We compare HRAG against a fine-tuned end-to-end T5 model (E2E T5), equivalent to the "*Naive*" model shown by Kale and Rastogi (2020), which achieved state-of-the-art on the MultiWoZ dataset as well as in the recent WebNLG Challenge 2020 (Castro Ferreira et al., 2020). We employ *t5-small* for the underlying PLMs of both HRAG and E2E T5, to be consistent with Kale and Rastogi (2020).

4 Results

4.1 Ablation Study

First, we present an ablation study of HRAG on the 5-shot SGD dataset aimed to analyse the impact of its components; the results are presented in Table 1. To examine the output of the lexicalisation module without aggregation, we simply concatenate the independently generated sub-phrases to form a unified text. As is to be expected, such a concatenation achieves low BLEU and BLEURT scores, clearly indicating the need for more sophisticated aggregation. Nevertheless, the lexicalisation module achieves 0% missing slot error thanks to its focus on individual units of information.

	BLEU ↑	BLEURT ↑	MER ↓
Lexicalisation	46.29	-0.39	**0.00**
+ aggregation	46.60	-0.30	1.16
+ post-edit	53.00	**-0.20**	1.13
+ selection	**53.04**	**-0.20**	0.14
E2E T5	50.15	-0.23	0.84
+ delex	50.25	-0.27	0.81

Table 1: Results of ablation study on 5-Shot SGD.

For the aggregation module, we examine the output of its final iteration. Its performance is on par with lexicalisation output, seemingly suggesting that aggregation offers little improvement. However, based on output analysis, the low BLEU and BLEURT are misleading and do not reflect the output quality. We attribute the lack of automatic score improvements to the module's tendency to overgenerate at the end of the output in anticipation of the next sub-phrase (as shown in the example in Figure 2). Other errors emerge from no-value attributes and due to sub-optimal training targets. MER increases the most during aggregation, as its recurrent nature is prone to error propagation. We should also note that using a single aggregation layer to aggregate all sub-phrases at the same time had comparable BLEU and BLEURT performance but underperformed by 8.18 points in MER.

The role of the post-edit module is to obviate errors propagated from the lexicalisation and aggregation modules, and it greatly improves performance by 6.4 and 0.1 points in BLEU and BLEURT respectively. Specifically, this stage fixes the lexicalisation of no-value attributes, removes overgenerated tokens, improves fluency, and adds or removes values that have been missed or repeated. Nonetheless, as this is an extra generation step, it occasionally removes some required values, as indicated by the almost constant slot error.

Due to these frequent imperfections in the post-

[2]Evaluation scripts will be released along with the code.

edit layer's output, the final output of HRAG is selected between the output of the last aggregation iteration and the output of the post-edit module according to which one has the lower MER. This process leads to the highest BLEU and BLEURT scores and a MER close to 0%.

Finally, we examine delexicalisation's impact on E2E T5, applying it similarly to how it is applied to HRAG. While HRAG benefits from delexicalisation as it improves its generalisation ability and helps reduce MER, we observe marginal improvements (a MER decrease of 0.03%) when applied over a strong end-to-end model like E2E T5.

4.2 Few-Shot Evaluation

Tables 2 show automatic evaluation results for E2E T5 and HRAG systems trained on an increasing amount of data on FewShotSGD, FewShotWeb and MultiWoZ datasets. Overall the behaviour of the two systems is consistent across the three datasets.

As discussed in Section 4.1, HRAG manages to preserve the input MR values throughout the generation, and as such outperforms E2E T5 in MER across all dataset splits by a significant margin, especially when trained on smaller splits. E2E T5 only overperforms on 0.1% MultiWoZ, but a closer examination of the outputs reveals that the 4.85% MER is achieved at great expense to fluency as the system simply copies all input MRs instead of generating utterances. Although HRAG was not completely unaffected by such behaviour, it was still able to generate relevant outputs thanks to its ability to independently lexicalise smaller and simpler sub-phrases which lead to improvements of 10.79 BLEU and 0.55 BLEURT scores over E2E T5 despite the higher MER on 0.1% MultiWoZ.

Results in Table 2 demonstrate the effectiveness of HRAG in extremely low-resourced conditions with differences in BLEU and BLEURT scores of 2.89 and 0.3 for FewShotSGD and 6.54 and 0.12 for FewShotWeb on their respective smaller splits. Improvements over the end-to-end systems converge as the number of training examples increases, but HRAG consistently performs best in MER across all datasets and training pool sizes. In terms of BLEU/BLEURT, HRAG is able to maintain an edge over E2E T5 on all FewShotSGD and FewShotWeb splits, while on MultiWoZ, E2E T5 appears to be the best performing system, especially in terms of BLEURT score with a difference up to 0.9. By looking at the system's outputs, however, HRAG appears to perform comparably or even outperform E2E T5 despite lower BLEURT scores; examples are provided in Appendix C.

Table 3 shows results on FewShotWoZ; models are trained and tested separately on each domain. Similarly to the results shown in Table 2, HRAG excels in terms of MER, missing on average 5% fewer values compared to E2E T5. In BLEU and BLEURT scores, while on average E2E T5 outperforms HRAG, there is no consistently better system. Unfortunately, only one reference per MR is provided making multi-reference scoring impossible and in extension BLEU more unreliable. In Section 4.4, we perform human evaluation to better assess

BLEU ↑	5	10	20	40	80
E2E T5	50.15	55.75	60.37	**62.53**	63.62
HRAG	**53.04**	**56.95**	**60.94**	62.49	**63.97**
BLEURT ↑	5	10	20	40	80
E2E T5	-0.23	-0.15	**-0.09**	**-0.06**	-0.05
HRAG	**-0.20**	**-0.13**	**-0.09**	**-0.06**	**-0.04**
MER ↓	5	10	20	40	80
E2E T5	0.84	0.65	0.37	0.34	0.27
HRAG	**0.14**	**0.05**	**0.03**	**0.07**	**0.01**

(a) FewShotSGD

BLEU ↑	1	2	3	4	5	6	7
E2E T5	21.46	37.47	41.17	45.31	45.09	45.42	46.40
HRAG	**28.00**	**39.04**	**43.89**	**45.64**	**45.61**	**45.62**	**46.68**
BLEURT ↑	1	2	3	4	5	6	7
E2E T5	-0.32	0.08	0.13	0.22	0.21	0.22	0.23
HRAG	**-0.20**	**0.11**	**0.19**	**0.24**	**0.23**	**0.23**	**0.25**
MER ↓	1	2	3	4	5	6	7
E2E T5	22.81	23.80	19.48	19.32	20.72	20.10	19.54
HRAG	**8.21**	**5.58**	**1.75**	**0.98**	**0.52**	**0.24**	**0.35**

(b) FewShotWeb

BLEU ↑	0.1%	0.5%	1%	5%	10%	20%
E2E T5	3.34	25.90	**41.27**	**48.77**	**50.65**	**52.56**
HRAG	**14.13**	**31.69**	40.39	48.72	49.71	50.34
BLEURT ↑	0.1%	0.5%	1%	5%	10%	20%
E2E T5	-1.29	-0.39	**-0.16**	**-0.08**	**-0.07**	**0.00**
HRAG	**-0.74**	**-0.33**	-0.18	-0.12	-0.10	-0.09
MER ↓	0.1%	0.5%	1%	5%	10%	20%
E2E T5	**4.85**	5.79	5.76	2.86	2.44	2.10
HRAG	7.45	**3.53**	**1.64**	**0.75**	**0.70**	**0.86**

(c) Reduced MultiWoZ

Table 2: Automatic evaluation results.

BLEU ↑	Restaurant	Laptop	Taxi	Tv	Train	Hotel	Attraction	AVG
E2E T5	**25.73**	**25.94**	17.62	**26.25**	15.04	28.34	**19.41**	**22.62**
HRAG	25.55	22.95	**18.55**	24.70	**19.31**	**29.96**	14.44	22.21
BLEURT ↑	**Restaurant**	**Laptop**	**Taxi**	**Tv**	**Train**	**Hotel**	**Attraction**	**AVG**
E2E T5	**-0.08**	**0.03**	-0.43	**0.02**	**-0.32**	-0.07	**-0.44**	**-0.18**
HRAG	-0.12	-0.11	**-0.40**	-0.09	-0.34	**-0.04**	-0.47	-0.22
MER ↓	**Restaurant**	**Laptop**	**Taxi**	**Tv**	**Train**	**Hotel**	**Attraction**	**AVG**
E2E T5	7.43	6.73	16.87	**3.80**	18.76	3.75	21.41	11.25
HRAG	**4.21**	**2.58**	**7.23**	4.15	**13.47**	**1.39**	**9.20**	**6.03**

Table 3: Automatic evaluation results on FewShotWoZ. AVG is the macro-average score across all domains.

systems performance on FewShotWoZ.

4.3 Zero-Shot Evaluation

We perform zero-shot analysis on SGD and WebNLG testsets; Figure 4 shows the results of the systems presented in Section 4.2 with reported performances split into domains seen and unseen during training according to the original datasets.[3]

In both datasets, HRAG achieves MER in unseen cases lower than even E2E T5's seen scores, further validating the generalisation ability of HRAG when little to no resources are available. Overall, HRAG achieves higher BLEU and BLUERT scores than E2E T5 as well, with the exception of BLEURT scores for FewShotSGD. However, similarly to what has been found in Section 4.2, HRAG's outputs do not appear to necessarily be more disfluent than E2E T5 outputs.

Interestingly, HRAG's MER for unseen FewShotWeb is lower than the corresponding seen one. We observe that HRAG tends to avoid generating complex sentence structures when dealing with unseen inputs, and simply concatenates the lexicalisation sub-phrases (e.g. "liselotte grschebina, born in the german empire, attended the school of applied arts in stuttgart, israel."). This strategy benefits FewShotWeb as HRAG focuses on copying elements from the input and effectively avoids introducing noise. Such behaviour is not observed for FewShotSGD, but seen/unseen MER are still comparable and in close proximity to 0%.

4.4 Human Evaluation

To account for the shortcomings of automatic evaluation, we employed the human evaluation framework Direct Assessment (Graham et al., 2017) to set up tasks on the Amazon Mechanical Turk

[3]Full results tables are shown in Appendix D.

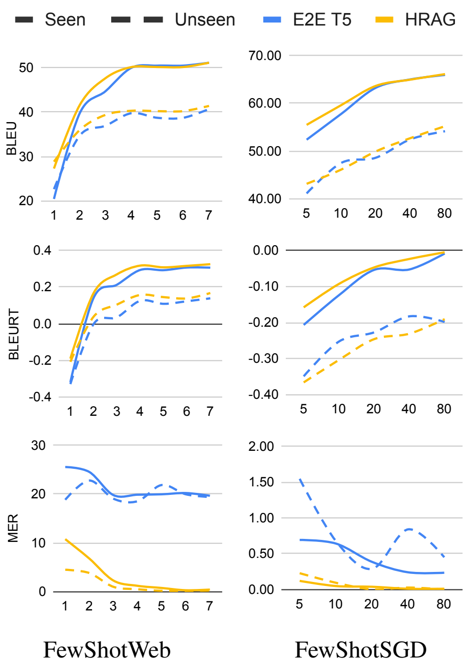

Figure 4: Zero-shot automatic evaluation results.

(AMT) platform and assess the fluency and adequacy of various models' outputs. We created separate tasks to assess the fluency and adequacy of the texts on two distinct subsets, in order to minimise correlation between the criteria. Specifically, we sampled 750 MRs from each test set of 5-shot SGD and FewShotSGD, and collected the corresponding outputs of HRAG, E2E T5, and the ground truth (GOLD); we include the latter to provide context to the evaluation. We picked the 5-shot subset of SGD to observe how the systems behave when exposed to the least amount of in-domain data. The pool of crowd-workers was limited to those residing in English-speaking countries, and who had a high acceptance rate; every text was evaluated by at least 3 crowd-workers on a 1 to 100 Likert scale. After consulting the crowd-workers' reliability based on

		Fluency		Adequacy	
		raw	z-score	raw	z-score
5S-SGD	GOLD	80.502	0.103	78.690	0.044
	E2E T5	76.355	-0.033	76.864	-0.017
	HRAG	**77.245**	**-0.012**	**77.508**	**0.041**
FS-WOZ	GOLD	76.936	0.018	80.210	0.066
	E2E T5	**75.845***	**0.016***	78.609	0.042
	HRAG	**75.824***	**0.014***	**79.096**	**0.043**

Table 4: Human Evaluation results; * denotes no statistically significant difference between assessments.

the Direct Assessment platform analysis, we had to filter out 39.5% of the participants.

Table 4 gathers the raw and mean standardised z-scores of the evaluation. Both models of course are considered worse than the ground truth, but HRAG performs better than E2E T5 in both fluency and adequacy, with the exception of fluency in FewShotWoZ where the systems exhibit no statistically significant difference (according to Wilcoxon rank-sum tests). These results further support the efficacy of HARG for few-shot settings.

5 Related work

Despite being an important research topic with real-life applications, domain adaptation for low-resource/few-shot concept-to-text NLG has not been extensively researched. Wen et al. (2016) leveraged the scarcity of target in-domain data by augmenting it with synthetic data, Tran and Nguyen (2018) used variational autoencoders in conjunction with text similarity and domain critics to better guide the fine-tuning process, while Mi et al. (2019) tackled the problem by defining domain adaptation as an optimisation meta-learning task. Most recently, Peng et al. (2020) and Kale and Rastogi (2020) have proposed the use of pretrained language models to tackle few-shot and zero-shot learning in concept-to-text NLG, achieving significant gains over strong non-pretrained baselines. Specifically, Peng et al. (2020) proposed SC-GPT, a semantically conditioned GPT-2 model, wherein, prior to few-shot learning, the GPT-2 model is further fine-tuned on a number of task-oriented dialogue datasets in order to mitigate the problem of representation bias. On the other hand, in Kale and Rastogi (2020), a set of human-authored templates are used to generate high-quality sentences corresponding to each unit of information in an MR. These are then concatenated and given as input to a T5 model (T2G2) to form a coherent sentence. In this paper's evaluation, we opt to compare our approach against the naive T5 baseline introduced by Kale and Rastogi (2020), as it is shown to overly outperform SC-GPT by basically replacing the underlying GPT-2 model for T5, and SC-GPT was outperform all previous non-pretrained baselines. We do not compare against T2G2, as access to human authored templates or other such manually annotated resources, which are by nature very domain-specific and costly to create, are not necessarily guaranteed in low-resource settings. We note that T2G2 is equivalent to the naive T5 when templates are not employed.

In our proposed system, the hierarchy emerges from modelling the lexicalisation and aggregation sub-tasks on separate layers. Previous attempts in exploring hierarchical structures for text generation tasks instead focused on modelling different aspects of the input or output. In concept-to-text NLG for task-oriented dialogues, Su et al. (2018) proposed a multi-layered decoding process where each layer was responsible for generating words associated with specific part-of-speech tags. Chen et al. (2019) and Tseng et al. (2019) took advantage of the intrinsically hierarchical structure of dialogue acts to create better input representations and ease domain adaptation. Our approach is also related to coarse-to-fine approaches, which have been explored in story (Fan et al., 2018), review (Li et al., 2019) and keyphrase (Chen et al., 2020) generation tasks. However, in these tasks, the output is not necessarily restricted to be an exact realisation of the input, and can be initially loosely prompted or drafted, and subsequently expanded.

6 Conclusion

We proposed Hierarchical Recurrent Aggregative Generation, a three-moduled jointly trained architecture, designed to exploit the different extents to which lexicalisation and aggregation can benefit from transfer learning. Due to the lack of explicit training signals for HRAG's modules, we also show how module-specific targets can be inferred from the available output targets. Extensive automatic metric experiments and analysis across 4 datasets, as well as accompanying human evaluation, shows that HRAG outperforms previous state-of-the-art, especially on missing slot error and adequacy.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Thiago Castro Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional WebNLG+ shared task: Overview and evaluation results (WebNLG+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*, pages 55–76, Dublin, Ireland (Virtual). Association for Computational Linguistics.

Wang Chen, Hou Pong Chan, Piji Li, and Irwin King. 2020. Exclusive hierarchical decoding for deep keyphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1095–1105, Online. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3696–3709, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Thiago Ferreira, Claire Gardent, Nikolai Ilinykh, Chris van der Lee, Simon Mille, Diego Moussallem, and Anastasia Shimorina. 2020. The 2020 bilingual, bi-directional webnlg+ shared task overview and evaluation results (webnlg+ 2020). In *Proceedings of the 3rd International Workshop on Natural Language Generation from the Semantic Web (WebNLG+)*.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. 2017. Can machine translation systems be evaluated by the crowd alone. *Natural Language Engineering*, 23(1):3–30.

Mihir Kale and Abhinav Rastogi. 2020. Template guided text generation for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6505–6520, Online. Association for Computational Linguistics.

Junyi Li, Wayne Xin Zhao, Ji-Rong Wen, and Yang Song. 2019. Generating long and informative reviews with aspect-aware coarse-to-fine decoding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1969–1979, Florence, Italy. Association for Computational Linguistics.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3151–3157. International Joint Conferences on Artificial Intelligence Organization.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

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. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 172–182, Online. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

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

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 4603–4611. PMLR.

Shang-Yu Su, Kai-Ling Lo, Yi-Ting Yeh, and Yun-Nung Chen. 2018. Natural language generation by hierarchical decoding with linguistic patterns. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 61–66, New Orleans, Louisiana. Association for Computational Linguistics.

Van-Khanh Tran and Le-Minh Nguyen. 2018. Adversarial domain adaptation for variational neural language generation in dialogue systems. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1205–1217, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Bo-Hsiang Tseng, PaweĹ, Budzianowski, Yen-chen Wu, and Milica Gasic. 2019. Tree-structured semantic encoder with knowledge sharing for domain adaptation in natural language generation. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 155–164, Stockholm, Sweden. Association for Computational Linguistics.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. MultiWOZ 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117, Online. Association for Computational Linguistics.

Giulio Zhou and Gerasimos Lampouras. 2021. Generalising multilingual concept-to-text NLG with language agnostic delexicalisation. 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 114–127, Online. Association for Computational Linguistics.

Barret Zoph, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. Transfer learning for low-resource neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1575, Austin, Texas. Association for Computational Linguistics.

A WebNLG few-shot splits

Table 5 details how many of the total data were kept in our WebNLG 3.0 few-shot splits (FewShotWeb); as the triple length grows, most property combinations are unique which results in a bigger portion of the data being included. Interestingly, the 1-triple subset covers 346 out of 372 occurring properties, which makes it particularly suited for supervised learning of our lexicalisation module.

B Configurations

Fine-tuning is performed with Adafactor (Shazeer and Stern, 2018) as an optimiser, with learning rate set to $1e^{-3}$ and Huggingface (Wolf et al., 2020)'s default parameters; gradient accumulation is used with a batch size of 256 for all the datasets except FewShotWoZ where the batch size is set to 1 as in (Peng et al., 2020); early-stopping is adopted with patience set to 30 and a combined loss between BLEU and slot error rate as the scoring function.

Reranking is performed as described in Section 2.5, with 5 lexicalisation and aggregation hypotheses generated at each time step. However, at training time, for computational reasons, only the lexicalisation outputs are reranked. At inference time, reranking is performed for both the baseline and HRAG's post-edit module, with 10 hypotheses generated and reranked. Each system is fine-tuned with 5 different seeds. Section 4 reports the average performance of each system.

C Examples

Table 6 shows examples of outputs produced by systems trained on 20% MultiWoZ where BLEURT score does not correlate.

Table 7 shows examples from FewShotWoZ where E2E T5 suffers from hallucinations.

D Full results

Tables 8 and 9 show the full results presented in Section 4.3.

E Inferred training signal examples

Tables 10, 11, 12, and 13 show examples of inferred training signals for the lexicalization and aggregation modules as discussed in Section 2.6. Note that as described in the aforementioned section, during training the input facts are ordered according to their values' appearance in the reference to facilitate the proper inference of the training signal for aggregation. Also note that we show the values intact in these examples for clarity, even though throughout our experiments the signal is inferred after a delexicalization pre-processing step.

# Triples	Few-shot data	Full data
1-triple	346	7686
2-triple	619	6948
3-triple	813	7610
4-triple	898	7061
5-triple	704	5084
6-triple	191	536
7-triple	168	501

Table 5: WebNLG 3.0 few-shot splits

MR:	hotel [inform (stars = 4)]	BLEURT
E2E T5:	it has a 4 star rating .	0.21
HRAG:	it ’s a 4 star hotel .	-0.79
MR:	attraction [request (name) | inform (area = city centre ; choice-1 = many]	BLEURT
E2E T5:	there are many attractions in the city centre . what is the name of the attraction you are looking for ?	0.01
HRAG:	there are many attractions in the city centre . do you have a name ?	-0.37
MR:	train [offerbook (none) | inform (leave = 05:59 ; arrive-1 = 07:27 ; depart = cambridge ; dest-1 = london ; dest-2 = liverpool street ; day = saturday ; id-1 = tr2895)]	BLEURT
E2E T5:	tr2895 leaves cambridge at 05:59 and arrives in london at 07:27 on saturday . would you like me to book you a ticket ?	0.05
HRAG:	i have tr2895 that leaves at 05:59 and arrives at 07:27 from cambridge to liverpool street london on saturday . would you like me to book it for you ?	-0.20

Table 6: 20% MultiWoZ output examples.

MR:	inform count (count = 11 ; near = nob hill ; type = restaurant)
E2E T5:	there are 11 restaurants near nob hill in the north hill area . it is a nice restaurant
HRAG:	there are 11 restaurants near nob hill area
MR:	inform (memory = 4 gb;; type = laptop ; name = satellite nereus 81 ; IsForBusiness = true)
E2E T5:	the satellite nereus 81 laptop is not for business computing and has 4 gb memory and a 4 gb memory
HRAG:	the satellite nereus 81 laptop has 4 gb of memory with a good battery rating , and is for business computing
MR:	inform (price = 10.10 gbp ; id = tr5433) | book (none)
E2E T5:	tr5433 will cost 10.10 gbp per ticket and leaves at tr5433 . can i book it for you ?
HRAG:	tr5433 price is 10.10 gbp payable at the station . can i book it for you ?

Table 7: FewShotWoZ output examples.

BLEU ↑	1	2	3	4	5	6	7
E2E T5	20.44	39.79	44.75	50.01	**50.48**	**51.06**	51.13
HRAG	**27.29**	**41.63**	**47.68**	**50.16**	50.17	50.40	**51.14**
BLEURT ↑	**1**	**2**	**3**	**4**	**5**	**6**	**7**
E2E T5	-0.32	0.14	0.21	0.29	0.29	0.31	0.31
HRAG	**-0.19**	**0.17**	**0.27**	**0.32**	**0.31**	**0.32**	**0.32**
MER ↓	**1**	**2**	**3**	**4**	**5**	**6**	**7**
E2E T5	25.54	24.52	19.78	19.86	19.95	20.21	19.65
HRAG	**10.76**	**6.78**	**2.28**	**1.20**	**0.76**	**0.29**	**0.43**

(a) Seen

BLEU ↑	1	2	3	4	5	6	7
E2E T5	22.65	34.74	36.95	39.77	38.73	38.73	40.70
HRAG	**28.84**	**35.99**	**39.40**	**40.31**	**40.23**	**40.23**	**41.41**
BLEURT ↑	**1**	**2**	**3**	**4**	**5**	**6**	**7**
E2E T5	-0.33	0.00	0.03	0.13	0.11	0.12	0.14
HRAG	**-0.21**	**0.04**	**0.10**	**0.16**	**0.14**	**0.14**	**0.17**
MER ↓	**1**	**2**	**3**	**4**	**5**	**6**	**7**
E2E T5	18.83	22.77	19.04	18.53	21.85	19.95	19.39
HRAG	**4.49**	**3.82**	**0.99**	**0.50**	**0.18**	**0.17**	**0.23**

(b) Unseen

Table 8: Full automatic evaluation results for FewShot-WoZ.

BLEU ↑	5	10	20	40	80
E2E T5	52.37	57.78	63.26	**65.01**	65.94
HRAG	**55.47**	**59.61**	**63.64**	64.96	**66.11**
BLEURT ↑	**5**	**10**	**20**	**40**	**80**
E2E T5	-0.21	-0.12	**-0.05**	-0.05	**-0.01**
HRAG	**-0.16**	**-0.09**	**-0.05**	**-0.02**	**-0.01**
MER ↓	**5**	**10**	**20**	**40**	**80**
E2E T 5	0.69	0.64	0.39	0.24	0.23
HRAG	**0.12**	**0.05**	**0.04**	**0.01**	**0.01**

(a) Seen

BLEU ↑	5	10	20	40	80
E2E T5	41.10	**47.50**	48.59	52.46	54.14
HRAG	**43.13**	46.11	**49.91**	**52.70**	**55.20**
BLEURT ↑	**5**	**10**	**20**	**40**	**80**
E2E T5	**-0.35**	**-0.25**	**-0.23**	**-0.18**	-0.20
HRAG	-0.37	-0.30	-0.25	-0.23	**-0.19**
MER ↓	**5**	**10**	**20**	**40**	**80**
E2E T5	1.55	0.68	0.29	0.84	0.45
HRAG	**0.23**	**0.09**	**0.00**	**0.03**	**0.00**

(b) Unseen

Table 9: Full automatic evaluation results for Few-ShotSGD.

MR: offer (pickup location = santa fe depot ; pickup date = march 2nd ; type = standard ; car name = accord) **Reference:** there is an accord , standard , at santa fe depot on march 2nd .	
Fact	**Inferred sub-phrase target**
1: offer (car name = accord)	there is an *accord*
2: offer (type = standard)	*standard*
3: offer (pickup location = santa fe depot)	at *santa fe depot* on
4: offer (pickup date = march 2nd)	on *march 2nd*
Facts to be combined	**Inferred aggregation target**
1 + 2	there is an *accord* , *standard*
1 + 2 + 3	there is an *accord* , *standard* , at *santa fe depot* on
1 + 2 + 3 + 4	there is an *accord* , *standard* , at *santa fe depot* on *march 2nd*
Post-edit target	
there is an accord , standard , at santa fe depot on march 2nd .	

MR: confirm (restaurant name = jo 's sushi bar ; location = pleasant hill ; time = 11 am ; number of seats = 2 ; date = march 13th) **Reference:** you want a table for 2 at jo 's sushi bar in pleasant hill at 11 am on march 13th ?	
Fact	**Inferred sub-phrase target**
1: confirm (number of seats = 2)	you want a table for *2* at
2: confirm (restaurant name = jo 's sushi bar)	at *jo 's sushi bar* in
3: confirm (location = pleasant hill)	in *pleasant hill* at
4: confirm (time = 11 am)	at *11 am* on
5: confirm (date = march 13th)	on *march 13th*
Facts to be combined	**Inferred aggregation target**
1 + 2	you want a table for *2* at *jo 's sushi bar* in
1 + 2 + 3	you want a table for *2* at *jo 's sushi bar* in *pleasant hill* at
1 + 2 + 3 + 4	you want a table for *2* at *jo 's sushi bar* in *pleasant hill* at *11 am* on
1 + 2 + 3 + 4 + 5	you want a table for *2* at *jo 's sushi bar* in *pleasant hill* at *11 am* on *march 13th*
Post-edit target	
you want a table for 2 at jo 's sushi bar in pleasant hill at 11 am on march 13th ?	

Table 10: Inferred training signal for the lexicalization and aggregation modules from FewShotSGD.

MR: inform_no_match (goodformeal = breakfast ; near = civic center)
Reference: unfortunately there are no restaurant -s near civic center that are good for breakfast

Fact	**Inferred sub-phrase target**
1: inform_no_match (near = civic center)	unfortunately there are no restaurant -s near *civic center* that are good for
2: inform_no_match (goodformeal = breakfast)	that are good for *breakfast*

Facts to be combined	**Inferred aggregation target**
1 + 2	unfortunately there are no restaurant -s near *civic center* that are good for *breakfast*

Post-edit target
unfortunately there are no restaurant -s near civic center that are good for breakfast

MR: inform (choice = 91 ; destination = cambridge ; departure = leicaster ; leaveat = the specified time), select (leaveat = 11:09 ; day = friday)
Reference: there are 91 trains leaving leicaster to cambridge after the specified time . how does friday 11:09 sound ?

Fact	**Inferred sub-phrase target**
1: inform (choice = 91)	there are *91* trains leaving
2: inform (departure = leicaster)	trains leaving *leicaster* to
3: inform (destination = cambridge)	to *cambridge* after
4: inform (leaveat = the specified time)	after *the specified time*
5: select (day = friday)	how does *friday*
6: select (leaveat = 11:09)	*11:09* sound

Facts to be combined	**Inferred aggregation target**
1 + 2	there are *91* trains leaving *leicaster* to
1 + 2 + 3	there are *91* trains leaving *leicaster* to *cambridge* after
1 + 2 + 3 + 4	there are *91* trains leaving *leicaster* to *cambridge* after *the specified time*
1 + 2 + 3 + 4 + 5	there are *91* trains leaving *leicaster* to *cambridge* after *the specified time* . how does *friday*
1 + 2 + 3 + 4 + 5 + 6	there are *91* trains leaving *leicaster* to *cambridge* after *the specified time* . how does *friday 11:09* sound

Post-edit target
there are 91 trains leaving leicaster to cambridge after the specified time . how does friday 11:09 sound ?

Table 11: Inferred training signal for the lexicalization and aggregation modules from FewShotWoZ.

MR: hotel_request (stars ; price ; area), hotel_inform (choice = 29)
Reference: there are 29 hotels that meet your needs . can you narrow it down to area , price range and stars ?

Fact	**Inferred sub-phrase target**
1: hotel_inform (choice = 29)	there are *29* hotels that meet your needs
2: hotel_request (area)	can you narrow it down to *area*
3: hotel_request (price)	*price* range and
4: hotel_request (stars)	range and *stars*

Facts to be combined	**Inferred aggregation target**
1 + 2	there are *29* hotels that meet your needs . can you narrow it down to *area*
1 + 2 + 3	there are *29* hotels that meet your needs . can you narrow it down to *area* , *price* range and
1 + 2 + 3 + 4	there are *29* hotels that meet your needs . can you narrow it down to *area* , *price* range and *stars*

Post-edit target
there are 29 hotels that meet your needs . can you narrow it down to area , price range and stars ?

MR: booking_nobook (time = 10:00 ; day = saturday), booking_request (time)
Reference: i am sorry we could not book you for saturday at 10:00 . would you like to try another time ?

Fact	**Inferred sub-phrase target**
1: booking_nobook (day = saturday)	i am sorry we could not book you for *saturday* at
2: booking_nobook (time = 10:00)	at *10:00*
3: booking_request (time)	you like to try another *time*

Facts to be combined	**Inferred aggregation target**
1 + 2	i am sorry we could not book you for *saturday* at *10:00*
1 + 2 + 3	i am sorry we could not book you for *saturday* at *10:00* . would you like to try another *time*

Post-edit target
i am sorry we could not book you for saturday at 10:00 . would you like to try another time ?

Table 12: Inferred training signal for the lexicalization and aggregation modules from reduced MultiWoZ.

MR: <amdavad ni gufa, location, ahmedabad> , <amdavad ni gufa, country, india> , <india, leader, sumitra mahajan>
Reference: amdavad ni gufa is located in ahmedabad , india , where sumitra mahajan is a leader .

Fact	**Inferred sub-phrase target**
1: <amdavad ni gufa, location, ahmedabad>	*amdavad ni gufa* is located in *ahmedabad*
2: <amdavad ni gufa, country, india>	*india*
3: <india, leader, sumitra mahajan>	where *sumitra mahajan* is a leader

Facts to be combined	**Inferred aggregation target**
1 + 2	*amdavad ni gufa* is located in *ahmedabad* , *india*
1 + 2 + 3	*amdavad ni gufa* is located in *ahmedabad* , *india* , where *sumitra mahajan* is a leader

Post-edit target
amdavad ni gufa is located in ahmedabad , india , where sumitra mahajan is a leader .

MR: <asterix (comics character), creator, rené goscinny> , <asterix (comics character), creator, albert uderzo>
Reference: the comic strip character asterix was created by albert uderzo and rene goscinny .

Fact	**Inferred sub-phrase target**
1: <asterix (comics character), creator, albert uderzo>	the comic strip character *asterix* was created by *albert uderzo* and
2: <asterix (comics character), creator, rené goscinny>	and *rene goscinny*

Facts to be combined	**Inferred aggregation target**
1 + 2	the comic strip character *asterix* was created by *albert uderzo* and *rene goscinny*

Post-edit target
the comic strip character asterix was created by albert uderzo and rene goscinny .

Table 13: Inferred training signal for the lexicalization and aggregation modules from FewShotWeb.