Fine-Grained Controllable Text Generation Using Non-Residual Prompting

Anonymous ACL submission

Abstract

The introduction of immensely large causal language models (CLMs) has rejuvenated the interest in open-ended text generation. However, controlling the generative process for these Transformer-based models is at large an unsolved problem. Earlier work has explored either plug-and-play decoding strategies or more powerful but blunt approaches such as prompting. There hence currently exists a trade-off between fine-grained control and the capability for more expressive high-level instructions. To alleviate this trade-off, we propose an encoder-decoder architecture that enables intermediate text prompts at arbitrary time steps. We propose a resource-efficient method for converting a pre-trained CLM into this architecture and demonstrate its potential in various experiments, including the novel task of contextualized word inclusion. Our method provides strong results in multiple experimental settings, proving itself to be both expressive and versatile.

1 Introduction

A causal language model (CLM) is a language model trained using a simple next-token prediction objective. Current CLMs are typically based on the Transformer architecture (Vaswani et al., 2017), which has resulted in unprecedented text generation capabilities (Radford et al., 2018a, 2019; Brown et al., 2020). Even so, the generation process of a CLM is difficult to control, as one is forced to gradually decode the next-step prediction one token at a time. This inhibits the applicability of CLMs when one intends for the generated text to fulfill certain criteria, and not only be a linguistically sound continuation to a given context.

Being able to control the text generation process is crucial for many real-world applications. As a straightforward example, we may want to control the generated text to counter the many biases that modern CLMs have been shown to posses (Bordia and Bowman, 2019). However, most applications require a greater degree of control, as one often wishes to steer the text generation towards a specific direction, such as generating a story to a given plot (Li et al., 2013; Yao et al., 2018; Riedl, 2021), or sticking to a certain topic (Keskar et al., 2019). Some areas require stringent and fine-grained control, as the many data-to-text tasks (Gardent et al., 2017; Leppänen et al., 2017; Koncel-Kedziorski et al., 2019), which necessitates that the generated text mediates very specific information and facts.

Due to this apparent need for controllable text generation, recent work (see Section 2.1) has explored different methods to steer and constrain the generation process of CLMs. They mainly fall into two lines of approaches. The more traditional line of approach focuses on fine-grained control and how to steer the generation process at arbitrary points, while still adhering to the current context. This is often achieved by independently modifying the predicted vocabulary distribution at each decoding step. However, this decouples the CLM from the control method, prohibiting the CLM's ability to plan accordingly and thus severely limits the type of control that can be formulated.

The second line of approaches has instead opted for more expressive and high-level control, having the CLM itself interpret and incorporate the instruction into the text generation. This is often done via either a fine-tuning objective, or as is currently common, by formulating the instruction as a textual context (referred to as prompting). Although expressive, these approaches are less effective than the previous one in controlling generation at specific points. For example, it is hard to use a prompt to generate a non-adjacent piece of text especially when the generation point is far away from the prompt. This is because the distance from the prompt and the next predicted token correlates negatively with the prompt's influence (Zou et al., 2021).

In an attempt to bridge the gap between fine-grained control and the expressiveness of prompts, we propose an architecture that allows for long-distance and independent prompting throughout the generation process. This architecture has an encoder-decoder setup, where the encoder influences the decoder via a novel non-residual attention schema. Along with theoretical arguments for the benefits of this architecture, we provide a resource-efficient self-supervised method for converting a pre-trained CLM into this setup.

In addition to evaluating on the original CommonGen dataset (Lin et al., 2020), we propose a new contextualized version of CommonGen, called Contextualised CommonGen (C^2GEN) and evaluate relevant methods on it. This new dataset extends the task to generating a sentence which includes a given set of words, while simultaneously adhering to a given context. We find no previous solution capable of this task, either barely including 50% of the target words, or not generating text of a satisfactory quality.

Our Contributions: (1) An encoder-decoder architecture based on a novel attention module which enables prompting at arbitrary time steps. (2) A resource-efficient method that requires no labeled data, for converting a pre-trained CLM into this architecture. (3) The introduction of the contextualized word inclusion task, through the C^2GEN dataset. (4) Extensive testing of related baselines and our proposed method, via both automatic and human evaluation.

2 Related Work

2.1 Controllable Text Generation

This section briefly introduces the related work for constrained text generation. A detailed description of each method, their strengths and weaknesses, and how they are configured to form our baselines is available in Appendix D.

Decoding strategies operate directly on the CLM's predicted vocabulary distribution at each time step, and are hence often model-agnostic. Dathathri et al. (2020) propose Plug-and-Play-Language-Models (PPLM) which adjusts the distribution in accordance with the gradients of an external discriminator model. Pascual et al. (2021) introduce Key2Text which steers the CLM to include target words by directly increasing their sampling probability, along with their GloVe (Pennington et al., 2014) neighbours.

Training objectives can also be setup to grant generative control, such as CTRL (Keskar et al., 2019) which incorporates control codes for textual genre. KG-BART (Liu et al., 2020) explicitly utilizes a common sense knowledge graph and fine-tunes towards word inclusion. GDC (Khalifa et al., 2021) fine-tunes towards arbitrary discriminator signal using Reinforcement Learning. POINTER (Zhang et al., 2020) tackles word inclusion constraints with a non-autoregressive approach, injecting words around the target words until a complete sentence is formed.

Prompting acts within the framework of the CLM's pre-training task, as constraints are expressed through natural language. This approach was popularized by the GPT models (Radford et al., 2018b; Brown et al., 2020) and has been shown to work for many different types of constraints (Reif et al., 2021; Clive et al., 2021).

2.2 Evaluation of Generated Text

There is no standardized evaluation methodology for open-ended text generation (Howcroft et al., 2020). The large number of possible good texts, hinders the usage of automatic text-overlap metrics (Papineni et al., 2002; Banerjee and Lavie, 2005; Lin, 2004). Human evaluation has also been reported to be problematic, as the vagueness of many surveys renders them irreproducible (Belz et al., 2020).

To remedy this, Van Der Lee et al. (2019) propose guidelines for human studies, and Gehrmann et al. (2021) strongly argue that textual quality cannot be described through a single metric. In order to conform to this discourse we investigate the properties of relevant methods in various situations with multiple metrics, without necessarily claiming one to be superior in all aspects.

3 Model Architecture

We propose to steer a CLM's generative direction by introducing a separate "encoder" for prompt instructions, which we refer to as the *prompt model*. The prompt model interprets textual prompts and produces positional invariant key-values, which the CLM can optionally attend to via a novel attention schema called *non-residual* attention (Section 3.1). The positional invariance ensures that the instruction is equally applicable at any time step, and is achieved by an additional shift of its key-values (Section 3.2).

Figure 1: Illustration of the non-residual attention during multiple time steps. The textual hidden states are shown in green, the non-residual in yellow, and the prompt model's in red. During the first time step both CLM streams self-attend to the input word, *I*, but the non-residual stream also attends to the prompt model's hidden states for the instruction *Very Happy*. At the second time step, *love* is inputted and both streams attend to the previous textual hidden states, and the non-residual stream again also attends to the prompt models instruction.

3.1 Non-Residual Attention

In order to allow for independent prompts at different time steps, we compute two independent streams of information for the CLM. These are referred to as the *textual* and *non-residual* streams. The textual stream ignores the prompt model completely, and is identical to the normal self-attentive forward pass of the CLM. The non-residual stream is responsible for the prediction at each time step, and is allowed to attend to both the previous steps of the textual stream, and potential key-values from the prompt model. This is depicted in Figure 1 and formalized in Equation 1.

Concretely, at time-step n the textual stream self-attends to the current time step and the previous textual key-values $KV_T^{i<n}$. The non-residual stream self-attends to the current time step, the previous textual key-values $KV_T^{i<n}$, and the prompt model's key-values KV_P. Finally, the next step prediction $P(w_{n+1})$ is computed from the non-residual stream.

Applying the prompt S_P to every time step in the text $S_{CLM} = \{w_1, w_2, :::, w_n\}$ thus results in:

$$\begin{aligned} KV_P &= \text{PromptModel}(S_P) \\ KV_T^n &= \text{CLM}(w_n \mid KV_T^{i<n}) \\ P(w_{n+1}) &= \text{CLM}(w_n \mid KV_p, KV_T^{i<n}) \end{aligned} \quad (1)$$

Non-residual key-values are hence never attended to by either streams from subsequent time steps. A prompt instruction at time step n can therefore only influence future decoding steps via the sampled token at time step n, and not through its key-values. This non-residual property of each prompt assures that the hidden state of the CLM does not deteriorate over time.

Intuitively, this assures that the residual key-values are only affected by textual input, allowing the CLM to operate within the limits of its pre-training objective. Furthermore, this means that one can apply different prompts at different time steps, without them disrupting each other through the CLM's internal state.[1] Further intuition of this architecture is available in Appendix C.

3.2 Position Invariant Transformation

Ideally, prompt instructions should be equally applicable at any time step in the generation process. However, the positional encoding system of Transformers makes this difficult, particularly absolute positional encodings (Vaswani et al., 2017). Overcoming this requires a significant amount of training of the prompt model (See Appendix C.2).

To alleviate the computational burden, we propose an architectural add-on where positional invariance is achieved by an additional set of weights, trained after the prompt model is trained on single sentence data. This reduces the overall training time, and allows one to easily fine-tune the prompt model on tasks lacking context, and apply the positional invariant transformation afterwards. This is depicted as step 3 an 4 in Figure 2.

The prompt model, being a CLM, uses causal self-attention to process text and generate sets of key-values per time step, where L refers to the number of layers in the model. We refer to the key-values at a time step i as kv^i. Hence, when the prompt model computes a prompt of length n, it yields the sequence of key-values $KV_P = \{kv^1, kv^2, :::, kv^n\}$.

[1] Ultimately, different prompts can be applied at each decoding step, but we imagine most use-cases will apply instructions on the sentence or paragraph level.

Figure 2: Overview of the resource-ef cient training procedure for creating a non-residual prompt model from a pre-trained CLM. Dashed lines indicate frozen weights. Hence, the weights of the CLM are frozen throughout all training steps and the prompt model's weights are frozen in step 3. During inference, the transformation learned in step 3 is inserted again.

The positional invariant transformation, referred to as C, consists of one parameter for each of the CLMs key-value parameters[2]. The same transformation C is then applied by point-wise addition to the prompt models output at at all time steps. Thus yielding the shifted key-values $KV_P = \{kv^1 + C;\ kv^2 + C;\ \ldots;\ kv^n + C\}$.

4 Training Procedure

Given a pre-trained CLM, we propose to train an accompanying prompt model via four distinct phases[3], as demonstrated in Figure 2. This includes an initialization phase, two pre-training phases, and one optional ne-tuning phase. The weights of the CLM are never updated in any of the training phases.

As popularized by Raffel et al. (2019), all training, independent of task, is formulated within the framework of teacher forced causal language modeling (Williams and Zipser, 1989). The goal is thus to maximize the likelihood of generating text S given prompt P, in accordance to Equation 1.

4.1 Initialization

Prior to any training, the prompt model is created by cloning the pre-trained CLM into a separate new model. The CLM and prompt model hence start with an identical set of weights. This results in an ef cient starting point, since the CLM is trained to communicate with itself via self-attention, and thus also the CLM and prompt model.

[2]The size of C hence depends on the model's number of layers, number of heads and its hidden size.

[3]This does not include the pre-training of the original CLM model.

4.2 Pre-training

Pre-training is divided up into two distinct phases, both relying on the text generation task of word inclusion with a target sentence length. In the rst phase, the prompt model is trained to in uence the CLM using only single sentence data, without any position invariant transformation. In the second phase only the position invariant transformation is learnt, by training on data with longer context. This is illustrated in Figure 3.

For both phases, training data is generated by sampling $[A; B]$ unique target words for each sentence $S = \{w_1; w_2; \ldots; w_n\}$, and incorporating them and the sentence length n into prompt P. The second phase utilizes sequences of multiple sentences, where each sentence is given its own prompt. During this phase, each prompt is computed independently, and the CLM attends only to the relevant prompt for each sentence.

Details regarding corpus and sampling schema used in our experiments is available in Appendix A, and details regarding our randomized prompt template is available in Appendix A.3.

4.3 Fine-tuning

Finally, one can optionally ne-tune the prompt model towards another task or dataset. This is done by temporarily removing the positional invariant transformation, and tuning only the prompt model. The positional invariant transformation is then re-inserted afterwards, shifting the now ne-tuned prompt model's key-values.

This ne-tuning schema circumvents the problem that many NLP tasks and labeled datasets are formulated without any accompanying context. And one can therefore utilize single sentence datasets, and still apply the prompt model at arbitrary time steps.

Figure 3: Detailed illustration of the two pre-training phases (Step 2 & 3 in Figure 2). Color indicates what prompt each generation step is affected by.

5 Contextualized CommonGen Dataset (C^2GEN)

CommonGen (Lin et al., 2020) is a dataset for the constrained text generation task of word inclusion. The objective of the task is to generate text that includes a given set of target words and adhering to common sense. Each sample includes 3-5 target words, taken from various image-caption datasets.

The samples in CommonGen are however all formulated without any accompanying context. We argue that this task formulation is too narrow, and that it needlessly incentivizes researchers to focus on methods that do not support context. Which is orthogonal to our belief that many application areas necessitates the consideration of surrounding context. Therefore, to complement CommonGen, we provide an extended test set where an additional context is provided for each set of target words. The task is therefore reformulated to both generate commonsensical text which include the given words, and also have the generated text adhere to the given context.

Each context is formulated as three sentences, created by human annotators from Mechanical Turk (www.mturk.com), as exempli ed in Table 1. The annotators were tasked to create three sentences, so that a subsequent sentence would be likely to include the target words. Details regarding the creation process of ^{2}**G**EN, and its statistical properties are available in Appendix F.

Jane was excited when the teacher announced it was career week. Jane signed her dad up to visit the classroom. On the appointed day, Jane's dad showed up dressed in his work gear.

Table 1: Example context from ^{2}GEN, where the target words for the subsequent sentence are day, repeman, retruck , front andtalk .

6 Word Inclusion Experiments

We separate word inclusion into two different settings. In the rst, the model is tasked to generate exactly 32 tokens. Requiring the model to both satisfy the word inclusion objective, and continue generating text according to the context. This allows methods that do not intuitively grant sentence level control to participate, such as PPLM and K2T.

In the second setting, the model is only tasked to create a single sentence, elevating the requirement of continued text generation. This setting is suitable for methods speci cally trained towards creating a single common sense sentence, such as KG-BART and POINTER.

For both of these settings, we run experiments on both CommonGen and ^{2}**G**EN. Since, experiments on the contextualized ^{2}GEN, require the model to adhere to a context regardless if the objective is to generate a single sentence or a free text, KG-BART and POINTER are excluded from these experiments all together.

6.1 Model Con gurations

Using our proposed method we train a non-residual prompt model to accompany a pre-trained GPT-2 Large model. This setup is referred to as NRP during experiments, and training details can be found in Appendix A. In order to demonstrate how more sophisticated decoding strategies can be incorporated, we also combine NRP with a slightly modi ed version of Key2Text. Details for this incorporation can be found in Appendix B.3.

The inference utilizes a beam size of 4, and any additional parameters were set according to a hold-out validation set (See Appendix B). All baseline implementations are taken from their respective code repositories, and if possible the of - cial pre-trained model (See Appendix D).

	Free Text (32 Tokens)				Single Sentence				
	Cov ↑	Ppl ↓	Self-Bleu ↓	Sense ↑	Cov ↑	Ppl ↓	Self-Bleu ↓	Sense ↑	Len
GPT-2 Large + Prompt	72.2	15.7	56.5	68.5	70.9	47.8	48.3	68.1	13.6
PPLM	13.3	17.2	21.6	77.9					
Key2Text	84.5	32.9	13.9	49.0					
POINTER					98.0	51.9	27.7	48.6	27.2
KG-BART					97.2	37.0	33.0	82.4	15.3
Our Contributions									
NRP	98.4	14.1	36.1	69.3	93.0	24.0	28.4	72.3	20.3
NRP + Key2Text	99.5	14.0	40.4	68.2	95.1	24.0	29.0	71.8	20.5

Table 2: Results for the Word Inclusion experiments on CommonGen.

	Free Text (32 Tokens)					Single Sentence					
	Cov ↑	Ppl ↓	Self-Bleu ↓	Sense ↑	Ctx ↑	Cov ↑	Ppl ↓	Self-Bleu ↓	Sense ↑	Ctx ↑	Len
GPT-2 Large + Prompt	57.0	12.5	31.7	81.2	76	56.6	24.9	31.7	88.0	74.3	13.6
PPLM	19.1	12.5	15.2	70.8	75.9						
Key2Text	93.9	18.0	15.4	56.5	76.4						
Our Contributions											
NRP	96.9	10.0	31.3	69.3	75.8	81.0	12.3	22.5	81.1	81.2	15.5
NRP + Key2Text	98.6	9.5	32.1	71.0	76.8	82.1	12.5	22.9	80.1	82.7	15.5

Table 3: Results for the Word Inclusion experiments on the β-GEN dataset.

6.2 Evaluation Metrics

In accordance to the guidelines described in Section 2.2, we provide both quantitative and qualitative evaluation. The qualitative examples in Table 4 are intended to convey the overall style for each algorithm, and more qualitative examples are available in Appendix G.

Quantitative metrics are easily comparable, but may be less suited to convey the overall style. Our quantitative metrics are described in detail in Appendix E, and brie y below:

Word Inclusion Coverage (Cov): The percentage of target words that are included in the generated text.

Perplexity (Ppl): The mean perplexity of the generated text according to language model, in this case GPT-2 XL. A lower perplexity indicates better language uency. In presence of contexts, as is the case with β-GEN, the perplexity is conditioned on the context.

Common Sense (Sense): The average score on how well the generated text adheres to common sense, according to human evaluators.

Self-BLEU-5 (Self-Bleu): Average BLEU-5 overlap between all generated texts. A lower score is desired as this indicates syntactic diversity.

Contextual Relevancy (Ctx): The average score on how well the generated text ts the given context, according to human evaluators.

6.3 Quantitative Results

We wish to highlight that NRP, Key2Text, and the prompted GPT-2 all control the same underlying CLM model. Differences between these approaches are hence a result of the method, not the model. Unfortunately, all quantitative metrics (including human metrics) are intrinsically correlated with sentence length, making comparisons of single sentences non-trivial (See Appendix E).

First, we note that only the NRP approaches, and arguably GPT-2, supports all four experiments. In general we nd that incorporating Key2Text with NRP increases the coverage slightly, but at the cost of a slightly higher self-Bleu. Hence, we refer to both of them as NRP throughout the remainder of this section.

In the CommonGen Free Text setting (Table 2), NRP achieves the best coverage rate by a large margin, and also the best perplexity. Noticeably, NRP outperform GPT-2 in all metrics, besides common sense where they are virtually equal. Interestingly, GPT-2's coverage is virtually the same as its free text counterpart, indicating that it quickly forgets the intended instruction. Key2Text generates the lowest self-Bleu, but has both the worst perplexity and common sense score. PPLM performs the best on common sense but instead fails the task completely, as demonstrated by its poor coverage.

NRP
The scooter riders wear a T-shirt that says "I Ride" on the back.
The player presses a button on the scanner to place the card in his or her inventory.
KG-BART
A man is riding a scooter and wearing a shirt.
A woman presses a button on a scanner and places a card on the scanner.
POINTER
you can ride: the scooter, a t t shirt, and then you have to wear a jacket.
by pressing a button, or pressing a card, the reader will place a button, and then press of the card into a scanner screen.

Table 4: Two independently generated sentences for NRP, KG-BART, and POINTER. The target words for the first sentence are: ride, scooter, shirt, wear, and the target words for the second sentence are: press, card, place, button, scanner.

In the CommonGen Single Sentence setting (Table 2), NRP fall slightly behind the specialized sentence methods in terms of coverage, but has a noticeably higher coverage than GPT-2. POINTER has the best coverage and self-Bleu, but also the worst common sense and dramatically worst perplexity. KG-BART has as expected the best common sense score, while staying fairly balanced on all other metrics. Again, NRP and GPT-2 show similar common sense scores.

In the C^2GEN Free Text setting (Table 3), NRP performs the best on coverage and perplexity. All methods perform nearly identical on the context score. Both PPLM and Key2Text perform better than they did on CommonGen, but Key2Text still has the worst perplexity and common sense, and PPLM still performs the worst on coverage. As expected, GPT-2 has a hard time focusing on word inclusion with the added context, resulting in its low coverage rate. This entails that GPT-2 acts more as a regular CLM ignoring its prompt, which results in the noticeably best common sense score.

Finally, NRP performs significantly better on coverage, perplexity, self-Bleu and context with Single Sentence on C^2GEN (Table 3). GPT-2 performs better on common sense, which is likely due to it focusing less on the word inclusion objective. Again, GPT-2 achieves a similar coverage as its Free Text counterpart.

6.4 Qualitative Results

As demonstrated in Table 4, NRP and GPT-2 tend to generate more linguistically complicated sentences, with more flow, compared to that of KG-BART. While stylistic complexity is arguably something desirable, it has the drawback that it increases the chance of generating text that breaks common sense. Our inspection also confirms that POINTER generates long sentences with weird formulations, that often break common sense. Examples of generated texts from the Free Text methods are available in Appendix G. Key2Text often inserts multiple line breaks, and sometimes gets stuck repeating a word. The difference between NRP, PPLM and GPT-2 is more subtle. The major distinction being that PPLM comes off as slightly more fluid in its formulations.

7 Sentence Length Experiments

The inclusion of sentence length in the pre-training objective (Section 4.2), gives an additional level of generative control over the linguistic style. As demonstrated in Table 5, the model incorporates and plans using the prompted sentence length, changing its wording accordingly.

We note that the model tends to prioritize textual quality over strictly sticking to the exact number of words. To measure this discrepancy, we generate sentences for all CommonGen validation samples for different prompted sentence lengths. Figure 4 shows the results from this experiment, displaying the expected offset for different prompted sentence lengths.

The mean offset is always above 0 and below 1, it is thus expected to generate a slightly longer sentence than intended. The standard deviation increases both as the prompted length approaches long, and short sentences. This matches the sentence distribution of the pre-training dataset, as demonstrated in Table 6 found in Appendix A.

L_P	L_G	Target Words	Generated Sentence
6	6	drink, sit, table, wine	The wine-drinkers sit on the table.
12	11	drink, sit, table, wine	The guests sit at a table and drink wine or beer.
18	16	drink, sit, table, wine	The guests sit at a table and drink wine , while the hostess sits on the oor.
6	7	soviet, china, invasion, people, news	The Soviet invasion of People's China news.
12	14	soviet, china, invasion, people, new	The news of the Soviet invasion was greeted with joy by people in China.
18	17	soviet, china, invasion, people, new	The news of the Soviet invasion in China was met with shock and dismay by many people.

Table 5: Examples of generated sentences for different prompted sentence lengths, using the pre-trained word inclusion model. L_p is the prompted length and L_G shows the resulting generation length.

8 Discussion & Future Work

We opted to demonstrate our architecture's capabilities on the task of word inclusion, since quantitative comparisons on this task is relatively straight-forward, compared to most other open-ended text generation tasks. Although, experimental results indicate the versatility of our approach, we cannot stress enough how our method conceptually generalizes to signi cantly more tasks.

Our non-residual architecture enables the use of prompt instructions at arbitrary time steps, and also hypothetically different types of prompts. We hence encourage future work to pursue the incorporation of multi-task prompt learning. As being able to apply exible prompts with precision would be a huge step forward in the many areas striving to utilize CLMs, such as generative story telling. Indeed, the ability to control the text generation process while considering context is key for any author tool intended for human editors.

Admittedly, the training method for realizing our encoder-decoder architecture has largely been dictated by a lack of resources. We conceptually prefer the more straight-forward training approach of training the prompt model directly on long context data. Something future work could investigate by increasing the computational budget, and investigation different positional encoding schemes.

Finally, we stress that nothing in our approach has focused explicitly on common sense. It is hence expected that methods that do, like KG-BART perform better on this metric. Future work could thus investigate the use of a prompt model to control a CLM ne-tuned towards common sense, or ne-tune a prompt model using common sense data. Results on CommonGen and C^2GEN dataset still have ample room for improvements.

9 Conclusion

This paper has introduced the concept of non-residual attention and demonstrated how it can be used to steer a generative text model. Additionally, our work pinpoints the lack of open-ended controllable text generation tasks that require the model to also account for a given context. We set out to remedy this by introducing the humanly created C^2GEN dataset, introducing the task of contextualized word inclusion.

Experimental results on C^2GEN and CommonGen, clearly demonstrates that using a non-residual prompt model increases generative control over a CLM. Compared to other methods, our approach stands out as the most versatile, consistently performing well across all tested situations. These results demonstrates the potential of our proposed architecture.

Figure 4: The generated sentence length offset from the prompted sentence length, on the CommonGen validation set. The curve illustrates the mean offset and the lled area shows the standard deviation.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. volume 9909, pages 382–398.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Anya Belz, Simon Mille, and David M. Howcroft. 2020. Disentangling the properties of human evaluation methods: A classi cation system to support comparability, meta-evaluation and reproducibility testing. In Proceedings of the 13th International Conference on Natural Language Generation, pages 183–194, Dublin, Ireland. Association for Computational Linguistics.

Shikha Bordia and Samuel R. Bowman. 2019. Identifying and reducing gender bias in word-level language models. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 7–15, Minneapolis, Minnesota. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. arXiv preprint arXiv:2006.14799.

Jordan Clive, Kris Cao, and Marek Rei. 2021. Control pre xes for text generation.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In International Conference on Learning Representations.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. Proceedings of the 10th International Conference on Natural Language Generation, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Aremu Anuoluwapo, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna Clinciu, Dipanjan Das, Kaustubh D Dhole, et al. 2021. The gem benchmark: Natural language generation, its evaluation and metrics. arXiv preprint arXiv:2102.01672.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In Proceedings of the 13th International Conference on Natural Language Generation, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. 2021. A distributional approach to controlled text generation. In International Conference on Learning Representations

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. 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 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. 2017. Data-driven news generation for automated journalism. Proceedings of the 10th International Conference on Natural Language Generation, pages 188–197, Santiago de Compostela, Spain. Association for Computational Linguistics.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark O. Riedl. 2013. Story generation with crowdsourced plot graphs. AAAI.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1823–1840, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pages 74–81.

Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2020. KG-BART: knowledge graph-augmented BART for generative commonsense reasoning. CoRR, abs/2009.12677.

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.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A plug-and-play method for controlled text generation. ArXiv, abs/2109.09707.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018a. Improving Language Understanding by Generative Pre-Training. Technical report, OpenAI.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018b. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a uni ed text-to-text transformer. CoRR, abs/1910.10683.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. 2021. A recipe for arbitrary text style transfer with large language models.

Mark Riedl. 2021. An introduction to ai story generation. The Gradient

Chris Van Der Lee, Albert Gatt, Emiel Van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In Proceedings of the 12th International Conference on Natural Language Generation, pages 355–368.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4566–4575.

Ronald J. Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. Neural Computation, 1(2):270–280.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2018. Plan-and-write: Towards better automatic storytelling. CoRR, abs/1811.05701.

Yizhe Zhang, Guoyin Wang, Chunyuan Li, Zhe Gan, Chris Brockett, and Bill Dolan. 2020. Pointer: Constrained progressive text generation via insertion-based generative pre-training. In EMNLP 2020

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '18, page 1097–1100, New York, NY, USA. Association for Computing Machinery.

Xu Zou, Da Yin, Qingyang Zhong, Ming Ding, Zhilin Yang, and Jie Tang. 2021. Controllable generation from pre-trained language models via inverse prompting.

Figure 5: Overview of the random prompt generation template. The left section illustrates the template formula for a single example. The right section demonstrates a full prompt with the target sentence length and the target words: Helmet, Motorcycle, Ride and Road. In this case the inclusion phrase was selected to be "should include", the separator sign ":", the delimiter "," , and the end sign ".".

A Training Details

A.1 Training Data

Both pre-training phases (step 2 & 3 in Figure 2) are based upon texts from Wikipedia. For the first pre-training phase, we only consider sentences where the number of tokens fulfills $5 \leq N_{tokens} \leq 32$. The length of the sentences remaining after this filtering are depicted in Figure 6. In the second pre-training phase, subsequent sentences are packed up until the combined token length reaches 128 tokens. The prompt model is then trained to instruct the CLM for each of the packed sentences made up of $5 \leq N_{tokens} \leq 32$ tokens. In both phases the number of tokens are given via the GPT-2 Tokenizer (Radford et al., 2018b).

Throughout both pre-training phases we sample $[A = 3; B = 6]$ unique words for each valid sentence, removing the probability of sampling stop words. If a sentence lacks 3 unique non-stop words it is removed from the first pre-training phase, and ignored during the second.

The reason for limiting the training corpus to sentences to a maximum of 32 tokens is to keep the computational burden low, in particular the maximum memory consumption. This is less of a problem in the second pre-training phase, where only the positional invariant transformation is trained, hence removing the need to store an optimizer state for the prompt model's parameters.

A.2 Training Settings

Both pre-training phases use the same set of hyperparameters. The batch size is set to 1280 samples. The maximum learning rate is set to 10^{-4}, following a linear warm up schedule for the first 500 update steps.

The training is performed with early stopping in regards to coverage, on a hold-out validation set. For the first pre-training phase, where the prompt model is trained towards single sentence data, this is done with the CommonGen validation set. In the second phase, a custom validation set created by sequences of Wikipedia sentences is used.

A.3 Randomized Prompt Template

In accordance to the popularized prompt paradigm of (Brown et al., 2020) we start each prompt with a couple of examples, followed by the actual instruction. To increase generalization during the pre-training we procedurally generate prompts where both the included examples and the overall formatting is randomized.

Each prompt include three examples which are uniformly sampled from a set of 50,000 sentences, that have been randomly selected and set aside from the pre-training dataset. The format is generated by joining the examples, target words and target sentence length, with a set of randomly selected combination tokens. These tokens are randomly selected from a fixed list of candidates. Example of a randomly generated prompt is available in Figure 5.

Figure 6: Distribution of sentence length in number of words for the filtered Wikipedia training data.

B Inference Details

B.1 Experiment Con gurations

All Non-residual prompt models utilize a repetition penalty of 1.25 and beam size of 4. For tasks without context they all start with the word "The". The prompted sentence length for each model is set via hyperparameter search on a held-out validation set. For CommonGen we use the provided validation set, and for the ELI5 dataset we use a portion of the pre-training dataset (Appendix A.1). The selected sentence lengths for each model and setting are available in Table 6.

B.2 Prompting Schema

All word inclusion experiments utilize the same simple inference schema. For each sample a single prompt instruction is generated, including all of the target words and sentence length. The CLM is then allowed to attend to this instruction using Non-residual attention, until all target words have been generated. After this the CLM continues entirety using the textual stream, and is thereafter identical to the original CLM. In the free text setting, this means that if the CLM ends a sentence without having included all target words, that the prompt instruction is still enabled. An example of this is illustrated in Figure 7.

We recognize that one could investigate a more ne-grained and active approach. For example, one could easily alter the prompt instruction to only cover words that are yet to be included, or extend the target sentence length if not all words are included towards the end. Although, we heavily encourage future work to investigate such approaches, The reason for our more simple approach is to lend more focus to the overall architectural contribution. This also demonstrates how one can easily steer the generation through high level instructions.

Model Name	Free Text	Sentence
NRP	10	15
NRP + Key2Text	10	15

Table 6: The prompted sentence lengths for both non-residual prompt models, for both the Free Text and Single Sentence setting.

Figure 7: Inference Prompting.

B.3 Key2Text Incorporation

To test the intuition of aiding the prompt model's high-level planning with direct decoding strategies we supply a slightly modi ed version of Key2Text(Pascual et al., 2021), that we nd to work better. This modi ed version is heavily inspired by the Max Only and No Guarantee version of Key2Text, as we only increase the probability of the target words and by not forcing them to appear. The major difference is that we not only increase the sampling probability for the target words, but also their different lemmas. Word lemmas are extracted from the target words via the use of Spacy (Honnibal and Montani, 2017).

The sampling probability modi er is applied in the same fashion as the repetition penalty of (Keskar et al., 2019), where a multiplier is applied on the logits values of the CLM. Target word that has been included have all of its lemmas effectively assigned a sampling probability of zero. Identical to the prompting schema described in Appendix B.2, this additional decoding strategy is deactivated once all the target words are included.

The sampling probability modi er for a non-included target word is given by Equation 2, where depicts the maximum increase, and the inclusion factor dictating the shape of exponentially increasing modi er curve. T depicts the fraction of completion for the generative process, in regards to the maximum number of generation steps. Hence, T is always within the range [0; 1].

$$1 + \frac{e^{T}}{e} \qquad (2)$$

All experiments in this paper utilize the same set of parameters, where = 0:5 and = 5:5. We note that increasing has the expected effect of increasing overall coverage, but we found this to be at a non acceptable loss of textual quality.

Figure 8: Illustration of the residual prompt effect that can happen in a Casual Language Model. The curved arrow indicates where the CLM should attend to the second instruction, but is also affected by the first instruction.

C Architectural Motivation

C.1 Residual vs Non-Residual Prompts

As explained in Section 3.1, the non-residual attention hinders an instruction at time step t_0 to influence future time steps via its key-values. If one instead applies prompts directly to the CLM's textual stream, it allows the prompt to have a lingering effect on future decoding steps, where it might not be desired. This is demonstrated in Figure 8, where the second instruction is the exact opposite of the first.

To what extent this effect actually occurs in large CLM's is left for future work.

C.2 Positional Invariant Transformation

The positional invariant transformation allows us to both reduce the maximum memory consumption during pre-training and cope with the absolute positional encoding system of GPT-2. Since we only train the positional invariant transformation during the second pre-training phase, the need to store an optimizer state for the prompt model's parameters is alleviated. In turn allowing us to increase the target sequence length without having to increase the computational memory load.

If one is not limited by computational resources, an intuitive alternative approach to tackling positional invariance is to actively change the positional encoding of the prompt model, to match the current generation step for the CLM. Unfortunately, we find that shifting the prompt model's absolute encodings is problematic in its own way, since this detrimentally impacts the pre-trained CLM's textual abilities. As demonstrated in Table 7, simply shifting the positional input encoding with a single step completely ruins the CLM's generative process.

We hypothesize that this positional frailty is due to the GPT-2's pre-training objective, which positionally encodes each input sequence from 0 and onward. These results leads us to speculate that directly converting a CLM with an absolute encoding system into a positional invariant prompt model, forces one to discard a significant portion of the information achieved during the CLM's pre-training. Hence, severely reducing the benefits of bootstrapping from a pre-trained CLM. Thus we estimate that such an approach entails both higher maximum memory consumption and also longer training time.

C.3 Cross-Attention vs Non-Residual Attention

In a traditional encoder-decoder architecture the encoder fully processes the input data, before any information is passed to the decoder. Admittedly, this makes intuitive sense, but it also entails that the decoder cannot do any computations before the encoder is completely finished. However, using a non-residual attention (or just regular self-attention) encoder, allows the encoder and decoder to executed in parallel. Effectively, increasing the theoretical inference speed by a factor of 2

Positional Encoding	Generated Text Continuation
[0; 1; 2; 3; 4; 5; 6; 7; 8]	"and she was walking along the path when she saw. . .
[1; 2; 3; 4; 5; 6; 7; 8; 9]	"was was was was was was was was was was was. . .
[8; 9; 10; 11; 12; 13; 14; 15; 16]	", , , , , , , , , , . . ".

Table 7: Generated text sequences continuing from the text "Emily was out walking in the park," encoded with different absolute positional encoding sequences. The model used is GPT-2 Large with a beam size of 1.

D Extended Related Work

D.1 Decoding Strategies

PPLM incorporates an attribute model (bag of words) in order to steer their CLM. The gradients from the attribute model pushes the activations within the CLM, increasing the probability of generating the target words. In CommonGen, the concept sets include only a few target words, while PPLM typically have been used with larger bags of words. This could potentially explain its poor coverage in our experiments. Furthermore, PPLM uses a GPT-2 Medium language model while NRP and K2T are based on the GPT-2 Large. We relied on the example BoW settings provided on the of cial GitHub page[4], and the target generation length was modi ed to 32 tokens.

PPLM incorporate a target generation length, but has no explicit sentence level control. Is thus omitted from all sentence-level experiments.

Key2Text directly increases the sampling probability of words semantically similar to the target words. It also provides a guarantee for word inclusion, enabling hard-constrained text generation. However, Key2Text is based on word stemming, while the coverage metric relies on lemmatization. This is likely the reason why Key2Text does not achieve 100% coverage in our experiments. We used the example ROC Story settings provided on the of cial GitHub page[5], and changed the generation length to 32 tokens.

Key2Text incorporate a target generation length, but yields no explicit sentence level control. Is thus omitted from all sentence-level experiments.

D.2 Training Strategies

CTRL incorporates genre-speci c control codes in its CLM pre-training objective. This way, the model learns to adapt its generation to t the target domain and genre. There is not any straightforward way of incorporating the word inclusion objective in this pre-training framework. Hence, we did not deem CTRL as a suitable baseline for our experiments.

KG-BART augments a sequence to sequence model (BART) with knowledge graphs to enhance its common sense reasoning abilities. It speci cally ne-tunes BART on the CommonGen training data, which is consists solely on single sentences. This results in the model generating short and succinct sentences, but not being capable continued text generation. Hence, KG-BART is omitted from the Free Text generation tasks. Moreover, KG-BART is designed to take a concept set as input, and not any additional context. It was hence removed from all experiments on the C^2GEN dataset.

We relied on the of cial scripts for ne-tuning and inference[6]. During inference KG-BART utilizes a beam search with beam size of 5, along with a ngram-based repetition constraint.

GDC propose controlling the text generation of pre-trained CLM by ne-tuning towards pointwise and distributional constraints with policy gradient. Additionally, GDC incorporates a KL-Divergence loss, to keep the ne-tuned model similar to the initial CLM, . However, this ne-tuning process is extraordinarily expensive requiring roughly 72 hours to tune a GPT-2 small model towards a single word. Furthermore, it lacks a setting for general word inclusion and is thus excluded from our experiments.

POINTER solves the word inclusion task in a non-autoregressive manner, iteratively injecting words around the target words. The target words are thus all included prior to the generation process. POINTER can therefore only fail on the coverage task if it combines a target word with another word. It is designed to generate on single sentences, and cannot handle contexts without signi cant modi cations. We used the standard decoding script provided on the of cial GitHub page[7] and changed the separator to ", ".

D.3 GPT Prompting

The currently most popular approach to steering CLM's is via what is called prompting. In this approach the desired instruction is formulated as a textual context, from which the CLM continues. Following the popular paradigm of few-shot prompting, each prompt starts with a number of examples of the word inclusion task. We found that careful tinkering of the prompt layout and the included examples had signi cant impact on the performance. The nal prompt used for the various experiments are available in Appendix A.3. A beam size of 4 was used during inference for all experiments.

[4] https://github.com/uber-research/PPLM
[5] https://github.com/dapascual/K2T
[6] https://github.com/yeliu918/KG-BART
[7] https://github.com/dreasysnail/POINTER

E Evaluation Metrics

E.1 String Matching As Evaluation Metrics

The current default way of evaluating CommonGen results includes the comparison of the generated sentences with a few target sentences using string matching metrics commonly seen in Machine Translation and Image Captioning. These include BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), CIDER (Vedantam et al., 2015), and (Anderson et al., 2016). We argue that the tasks of Machine Translation and Image Captioning impose harder constraints on the generated texts and could, therefore, more easily be compared to target sentences. However, in the case of controllable text generation with a few target words, there are vastly more room for creativity and thus difficult to compare to just a few target sentences. This means that a model with a particular writing style could fail to align with the target style, and achieve unjustified poor results. For this reason, we evaluate the generated texts differently, combining automatic metrics with human evaluation.

E.2 Automatic Metrics

Word Inclusion Coverage is, like in (Lin et al., 2020), calculated using lemmatization of the generated texts. However, due to the obvious flaws of comparing lemmatized output with non-lemmatized input, we also lemmatize the input words. Furthermore, we attempt to find all lemmas for each word and if no lemma is found for a given word, the word itself is instead used for comparison. The correlation between coverage and sequence length varies for models, but longer sequences often provide more space to include the target words.

Perplexity: We are calculating perplexity of the generated token sequence $X = (x_c; x_{c+1}; ...; x_{c+n})$, where c is the context length, which is 0 for the original CommonGen. The formula below is used with the pre-trained GPT-2 XL. This means, that for the C^{2}GEN dataset, the perplexity is conditioned on the context. The perplexity is typically strongly correlated with sequence length.

$$\exp\left\{ \frac{1}{n} \sum_{i=c+1}^{c+n} \log p \ (x_i \ j \ x_{<i}) \right\}$$

Self-BLEU-5 (Zhu et al., 2018) evaluates the syntactic diversity of a given set of texts. It is defined as the average BLEU-5 (Papineni et al., 2002) of each text, where each BLEU score is calculated by treating the current text as the hypothesis and the others as candidates. Longer sequences often yield lower BLEU scores.

E.3 Human Evaluation Metrics

In the experiment we take 100 random samples for each algorithm, where each sample is evaluated by 5 different people. We use Mturk to execute the experiments, where we request only native English speakers. In the experiment, we evaluate the generated text based on two specific criteria: Common Sense and Contextual Relevancy.

Figure 9 gives an example of the task. As we use Mturk we are unable to give a more thorough overview of the demographics and to control the order between the samples shown. But Mturk does use different annotators and divide the task into smaller tasks, automatically resulting in a varying demographic and random ordering of samples. On behalf of the inter-annotator agreement (IAA) we have a percent agreement of 0.51 and a Fleiss' of 0.115 (Celikyilmaz et al., 2020). In Table 8 we added the values for each of the experiments separately. In many cases we have a low Fleiss' and a mediocre percent agreement. The low Fleiss' can be explained by the fact that in all the received results 'Yes' is chosen in 70% of the samples. Fleiss' takes this distribution into account, that the agreement of two annotators is random. Secondly we see in Table 8 that IAA is lower for Free Texts experiments compaired with Sentence experiments. For transparency we will upload all the results of the human evaluation with the source code.

Common Sense: In the experiments the evaluators can answer No, Partly and Yes on the question whether the sentence makes sense. To construct the score presented in Tables 2 and 3 we map No to 0, Partly to 0.5 and Yes to 1. This allows us to create an average score of common sense where 1 is the upper bound and 0 is the lower bound.

Contextual Relevancy (CTX): Similar as above are the possible answers on the question whether the sentence is a logical continuation of the context: No, Partly and Yes. The No is mapped to 0, Partly to 0.5 and Yes to 1 resulting again in the scores depicted in the tables 2 and 3.

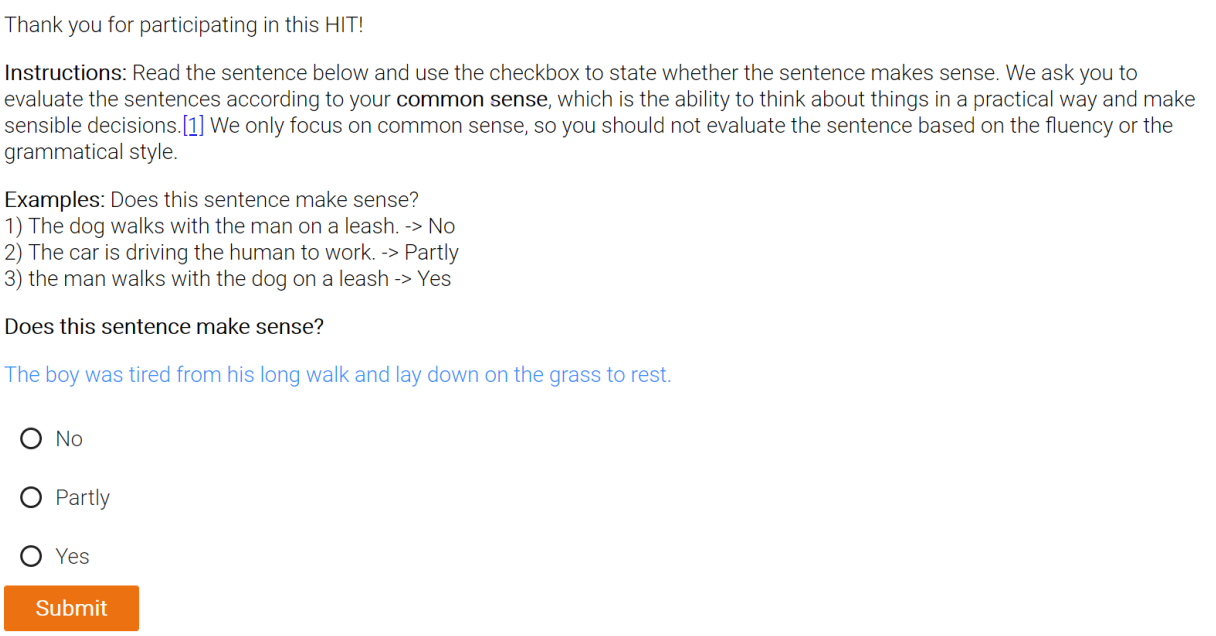

Figure 9: An example of a human intelligent task on Mturk. The evaluator is instructed to classify the level of common sense of a generated sentence.

CommonGen	**Free Text (32 Tokens)**		**Single Sentence**	
	Sense	Ctx	Sense	Ctx
GPT-2 Large + Prompt	0.001/0.411	-	0.356/0.653	-
PPLM	-0.024/0.493	-	-	-
Key2Text	0.021/0.358	-	-	-
POINTER	-	-	0.230/0.50	-
KG-BART	-	-	0.237/0.698	-
NRP	0.076/0.422	-	0.142/0.575	-
NRP + Key2Text	0.087/0.452	-	0.102/0.548	-
C^2GEN				
GPT-2 Large + Prompt	-0.003/0.552	-0.007/0.465	0.312/0.802	0.257/0.445
PPLM	0.023/0.493	0.018/0.477	-	-
Key2Text	-0.025/0.442	-0.021/0.463	-	-
NRP	0.049/0.438	-0.033/0.449	0.2073/0.693	-0.040.537
NRP + Key2Text	0.039/0.445	-0.029/0.462	0.219/0.688	-0.033/0.566

Table 8: The IAA of the human evaluation experiments. Each cell contains respectively the Fleiss' and percent agreement.

	Cov ↑	Ppl ↓	Self-Bleu ↓
GPT-2 Large Free Text (32 Tokens)	4.6	7.3	43.2
GPT-2 Large Single Sentence	3.8	7.5	37.9

Table 9: The results on the C^{2}GEN dataset with the GPT-2 Large model prompted with the context only, i.e. no prompted instruction on the target words. This provides a simple baseline on how likely the target words are to appear naturally, when the model only focuses on generating a natural continuation of the context.

F Dataset C^{2}GEN

The C^{2}GEN dataset is mainly constructed in accordance to the data in the CommonGen test set. However, minor modifications were made to the distribution of the number of word targets per sample. The CommonGen test set only consists of samples with 4 or 5 target words, whilst the training and validation data includes samples with only 3 target words. After careful reading we found no argument for this, and speculate it is purposefully intended to make the task more difficult. Since we disagree with artificially increasing task difficulty by skewing the test set away from the intended training data, we set attempted to remedy this in C^{2}GEN.

This was achieved by removing certain words from the CommonGen test set, resulting in that all set sizes are equally represented. Which in turn results in a more similar and fair distribution between C^{2}GEN, and the CommonGen train and dev split. Since, this filtering of words from the CommonGen test set results in duplicates, which weere removed, C^{2}GEN set is slightly smaller. However, as can be seen in Table 10, this discrepancy is insignificant.

Each remaining concept set manually received a textual context, in accordance to the explanation in Section 5. To assure the quality of the data generation, only native English speakers with a recorded high acceptance were allowed to participate. Finally, all contexts were manually verified, and fixed in terms of typos and poor.

The context lengths of the collected dataset is visualized in Figure 10. The perplexity and the Self-Blue metrics of the texts are available in Table 10. To evaluate the probability of the target words appearing naturally in a textual continuation, we use GPT-2 to generate text without the given target words and report on the coverage in Table 9.

Statistic	CommonGen	C^{2}GEN
# Concept-Sets	1,497	1,483
- Size = 3	-	494
- Size = 4	747	496
- Size = 5	750	493
# Unique Concepts	1,248	1,122
# Unique Concept-Pairs	8,777	6,835
# Unique Concept-Triples	9,920	6,959
% Unseen Concepts	8.97%	7.75%
% Unseen Concept-Pairs	100.00%	100.00%
% Unseen Concept-Triples	100.00%	100.00%
	C^{2}GEN	
Average context-sentence length	43.92	10.32
Perplexity (GPT-2 XL)	21.93	
Self-Bleu	16.63	

Table 10: A statistical comparison between the original CommonGen test dataset and the C^{2}GEN dataset.

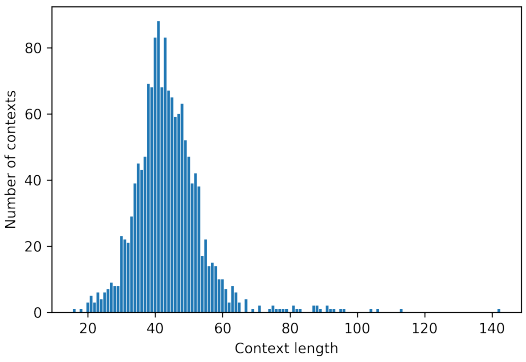

Figure 10: Distribution over number of words in the context of the C^{2}GEN.

G Qualitative Examples

	NRA
1	The players **sit** in **front** of a **microphone**, with the **guitar** and drums on one side of them.
2	The most common **use** of the **tool** is to remove a **piece** of **metal** or plastic from an object, such as a door knob.
3	The **dog walks** on a **leash**, and the owner can **walk** their pet on the **sidewalk** or in front of their home.
	GPT2-Large
1	The **guitar** was on the **front** of the sofa.
2	The **piece** of **metal** was **used** in the **tool**.
3	The **dog** was **walking** on the **sidewalk**.
	POINTER
1	he is shown **sitting** in **front** of a stage, playing his **guitar**, and standing behind a **microphone** on the one where he had to **sit** in the back.
2	, also known as a **metal** plate or, is a **piece** of **metal**, a sharp **tool**, that can be removed with the **use** of other **tools**.
3	this allows a **dog** that has been on a **leash** to be lowered onto the **sidewalk**, or to **walk** along the **sidewalk**.
	KG-BART
1	A man with a **guitar sits** in **front** of a **microphone** and plays his **guitar**.
2	A man **uses** a **tool** to remove a **piece** of **metal** from a **metal piece** of wood.
3	A **dog walks** on a **leash** on a **sidewalk**.
	PPLM
1	The new year is just around the corner, and with it comes a plethora of new releases and albums, but this year's lineup is the most impressive we've
2	The United States is one of only a few industrialized democracies which allow its citizens to buy and **use** drugs on behalf of their patients, and to provide them with a
3	The "Boomer" and "Boomer-Bom" are two words that mean very different things. When you say something to a person that
	Key2Text
1	Right **front** corner of the **guitar**.ՈnՈnՈnnSituation: You are holding a **microphone** in one hand while playing a **guitar**.ՈnՈnՈnnReason for selecting the
2	**Metal** Gear Solid V: The Phantom Pain **use** a **tool** called **Metal** Gear Solid V **Tools** to **piece** together the main storyline. **Metal** Gear Solid V: The Phantom Pain
3	Ad **leash** for **dogs**ՈnՈnՈnn**Dog walkers** and **dog walkers** with **sidewalk dog** kennelsՈnՈnՈnnThose of you that own or rent commercial **dog**

Table 11: Generated texts for 3 samples of the CommonGen Test set for the different methods. The inclusion words for the samples are the following; 1: [**front**, **guitar**, **microphone**, **sit**], 2: [**metal**, **piece**, **tool**, **use**], 3: [**dog**, **leash**, **sidewalk**, **walk**], highlighted in blue, brown, and green respectively. For models that are evaluated on both Free Text and Single Sentence, the sentence variant is shown.

H GPT Prompt Templates