Count Your Words Before They Hatch: Investigating Word Count Control

Stanford CS224N Custom Project

Katherine Li
Department of Computer Science
Stanford University
kathli@stanford.edu

Abstract

We present the method of *word count control* (WCC): a method of generating sentences of specified length using an autoregressive Transformer decoder model. We prepend special tokens representing the length of a sentence before the tokens representing that sentence in training samples, helping the model learn how to count words. We train from scratch a 125M parameter model on the WCC method, and find that the trained model significantly outperforms ChatGPT and GPT-2 large at generating sentences of a desired word count, despite the smaller size. We also find that training the model using the WCC method does not lead to any significant decrease in generated text quality. We analyze the ability of our model trained with WCC to generate sentences of varying lengths and find that it is more difficult to generate sentences of length 1 and of length higher than 50. Our findings indicate that transformer decoder models can plan words ahead even when generating one token at a time from left to right, when they are trained with an understanding of word count.

1 Key Information to include

- Mentor: Caleb Ziems

2 Introduction

Autoregressive, transformer-based language models have seen an explosion of popularity in recent years. Models like ChatGPT and GPT-4 are widely used for everyday purposes. While the abilities of these models are very impressive and seem to match human performance in many ways, there are also several failure cases of tasks that humans can perform fairly easily but these large language models (LLMs) seem to struggle with.

One notable example is the fact that LLMs seem to struggle with counting words. For example, as of March 2024, prompting ChatGPT with the instruction `"Generate me a sentence 25 words long about advice on how to train a dog."` yields the 24 word sentence `"Consistency is key in dog training; use positive reinforcement techniques, establish clear boundaries, maintain patience, and ensure regular exercise for a well-behaved, happy pup."`

The ability to count words has many applications. A user may want generated text to stay under a certain word count when generating a response, summary, or abstract. A user might want to generate poetry or creative writing that has a specific structure that requires a certain word count. A user attempting to evaluate an LLM with generated text from another model might want the generated text to have a certain length distribution!

Intuitively, the reason why LLMs perform poorly at generating text of a certain word count seems to be a combination of their autoregressive nature and tokenization. Specifically, transformer models are built on the concept of left-to-right generation where tokens are generated one at a time from left to right, but in order to generate a sentence with a certain length a model must be able to plan out more of the sentence than just the next word to ensure that the sentence ends grammatically in the right word count (for example, even if "and" is the most likely next word, it might not be grammatically correct to end a sentence with "and"). Also, LLMs break text into tokens, where the boundaries of tokens may not uniformly be individual words, which may further increase the difficulty of counting the number of words generated.

In this work, we explore a controllable generation method to induce the ability of generating sentences of a certain word count in a transformer-based language model. We investigate the ability of transformer models to succeed at this task to shed some light on the ability of LLMs to plan what they are generating several words in advance.

Contribution. We introduce a novel controllable generation method, word count control (WCC), to give an LLM the ability to generate sentences of a desired length with high accuracy. When a transformer model of comparable size to GPT-2 small is trained on this method, it is able to beat significantly larger models by a significant margin on the task of generating sentences of a desired length.

3 Related Work

Controllable generation. The field of controllable generation studies the ability of language models to meet controllable constraints at the wishes of human users. Controllable generation problems have a general framework of three parts: a controlled condition, a generative model, and generated output satisfying the input control condition. Zhang et al. (2022) provides a survey of common conditions in controllable text generation, which include topic, emotion, keywords, dialogue, persona, intent, storytelling, data-to-text, data augmentation, debiasing, and format control.

The method of prepending an WCC token to the sentence described by that token shares some similarities with the method of control codes used in CTRL (Keskar et al., 2019). In CTRL, control codes that specify the overall style of generated text are prepended to the beginning of training samples. These control codes mostly specify the domain of training data, such as "Wikipedia," "Books," or "Reviews." This method of control codes has been used for several applications, including in Spangher et al. (2022) to generate documents exhibiting narrative structure.

Numerical planning evaluation. Sun et al. (2023) evaluates several LLMs on various controllable generation tasks, one of which is numerical planning, where the task is to generate text satisfying constraints on the number of words, syllables, sentences, or paragraphs. Overall, LLMs perform poorly at the numerical planning benchmark (NPB), despite it being fairly simple. Similarly, Kallini et al. (2024) investigates the ability of language models to learn "impossible" languages, including a language which requires the model to learn grammar rules that involve counting positions of words or tokens. They find that LLMs are better at learning rules that involve counting words instead of counting tokens, which corroborates Sun et al.'s claim that LLMs have difficulty counting words.

Numerical control methods. To the best of our knowledge, our method is novel in injecting tokens representing the length of a sentence to perform word count control on transformer-decoder models. However, some related methods for controlling numerical properties of text include the hierarchical framework used in Tian et al. (2023) to generate song lyrics with the right syllable count to match a melody. Diffusion-LM (Li et al., 2022) presents a non-autoregressive, diffusion-based method of controlling text generation on several tasks, one of which includes generation length. InstructCTG (Zhou et al., 2023) finetunes a pretrained encoder-decoder model on several controllable generation tasks, one of which includes generation length.

4 Approach

We introduce an additional set of tokens to the GPT-2 tokenizer. We call these tokens *word count control* (WCC) tokens. Each WCC token `<N>` represents a sentence that is N words long.

During the tokenization step of training the model, we split the documents into individual sentences. We count up the number of words in each sentence and tokenize each sentence individually. We then prepend the WCC token corresponding to the word count for that sentence to the tokens that represent the words of the sentence. Specifically, the token <N> $= [\texttt{[EOS]} + 1 + N]$ represents a sentence with N words where $[\texttt{EOS}]$ is the end of sequence token that is usually the last token in the vocabulary. This means that the new vocabulary size for the model is the original vocabulary size plus the maximum sentence length N plus one. (The token $[\texttt{[EOS]} + 1]$ is used for the special case of a sentence that does not end within the context length, where the entire context length contains a single unfinished sentence.)

We then concatenate all of the tokenized sentences to create training samples with the right context length (in principle, this is similar to the idea of packing multiple documents into a single training sequence as described in Brown et al. (2020)).

To split the documents into sentences, we use Stanza (Qi et al., 2020), which includes a sentence segmentation tool that segments a document into individual sentences with reasonable accuracy.

As an implementation detail, some "sentences" in the dataset did not end in punctuation marks, often because they were titles or headers such as `== Family ==`. We discard all of these sentences from the dataset, since they are unsuited to the task of training the model to generate sentences of a desired length.

We compare the results of our method to the baseline performance of ChatGPT and a finetuned GPT-2 large model as reported in Sun et al. (2023).

5 Experiments

5.1 Data

We use the OpenWebText dataset (Gokaslan and Cohen, 2019), an open source recreation of OpenAI's WebText dataset (Radford et al., 2019) of English text data scraped from the internet.

5.2 Evaluation method

We use success rate (SR) and mean squared error (MSE) of the length of the generated sentence to evaluate the model. Specifically, the SR is the fraction of the sentences where the word count of the sentence matches the WCC token, and the MSE is taken between the "ground truth" length denoted by the WCC token and the actual generated sentence length.

Just like in processing the training samples, we use Stanza's sentence segmentation tool to segment the sentences.

5.3 Experimental details

We train from scratch one Transformer model with the GPT-2 architecture (Radford et al., 2019) using the approach described above. The model is of the same size as GPT-2 small (approximately 125 million parameters).

We use a context window size of 1024 tokens. As a result, the maximum value of N for a possible WCC token <N> is 1023, since 1023 tokens is the maximum number of words that can fit into the context window with an WCC token before. Therefore, because the original vocabulary size for the GPT-2 tokenizer is 50257, the new vocabulary size for the WCC model is 51281.

We train the model for 400,000 gradient steps for a total of 209 billion tokens. This is roughly 23 epochs of the dataset. We use cross-entropy loss and the AdamW optimizer with learning rate 0.0006, with a warmup of 4,000 steps and a single cycle cosine decay (Loshchilov and Hutter, 2016).

It takes roughly 3.5 days to train the model on a v3-32 TPU.

5.4 Results

Validation loss The model achieves a final validation loss of 2.908.

Generation quality Before discussing the results in word count control, we check that word count control does not significantly decrease the quality of the text the model is able to generate. We use the MAUVE score (Pillutla et al., 2021) to measure generation quality. At a high level, MAUVE is a measure of the divergence between the distribution of generated text from a model and the distribution of a reference target source of text. MAUVE provides a scalar summary of the trade-off between Type I error (the model placing high mass on text which is unlikely under the target distribution) and Type II error (the model being unable to generate text which is plausible under the target distribution).

We compute the MAUVE score between 5000 samples of generated text from the model with WCC and 5000 samples of text from OpenWebText. As a comparison, we also compute the MAUVE score between the base GPT-2 small model trained by OpenAI and OpenWebText. Since not all of the samples are uniformly the full context window (1024 tokens) long, we truncate all samples to the first 256 tokens. MAUVE scores, to two significant digits, are reported in Table 1. Notice that there is no significant difference between the two MAUVE scores, indicating that using the WCC method does not significantly impact generation quality.

Table 1: Comparison of MAUVE scores against OpenWebText

Model	**MAUVE score**
WCC model	0.54
Base GPT-2 small	0.54

5.4.1 SR and MSE

We evaluate the average SR and MSE over sentence lengths $N = 2$ to $N = 10$, with 100 samples per sentence length (following the sentence lengths and number of samples in Sun et al. (2023)) For each sample, we use a different prompt sentence and have the model generate the sentence after the prompt sentence. These results are reported in Table 2. Notice that our method significantly outperforms all of the baselines, including a ChatGPT and a finetuned GPT-2 large model, despite our model being significantly smaller.

Model	SR	MSE
GPT-2 large (fine-tuned)	0.64	1.62
ChatGPT	0.41	3.64
$\text{ChatGPT}_{\texttt{ICL}}$	0.37	4.95
WCC (ours)	**0.98**	**0.93**

Table 2: Comparison to baseline SR and MSE, reported over sentence lengths $N = 2$ to $N = 10$. Results that are not ours are from (Sun et al., 2023). We only report the results from GPT-2 large and ChatGPT, since they outperform the other models evaluated by Sun et al. Note that, as Sun et al. reports with surprise, in-context learning deteriorates the performance of models at generating sentences of given word count. (This is because the LLM tries to mimic the features, including length, of the in-context examples and are therefore more likely to generate outputs wit the wrong word count.)

To create prompt sentences that suggest the following text to be a plain English sentence, we prompted ChatGPT many times to generate sentences that suggested a topic for a following sentence. We then hand-picked 100 of the best prompt sentences to evaluate the model on. We tried to choose sentences that were grammatically correct but varied in sentence structure and topic. Details of how these sentences were generated are in Appendix A.1.

For a more thorough evaluation, we also evaluate the model on generating sentences of length from 1 to 100 (ignoring sentence lengths longer than 100 since they occur rarely in English text anyways). For each sentence length, we generate 100 evaluation samples . For the settings with prompt text, this is one sample per prompt. For each generation setting, we plot the SR and MSE (normalized by sentence length to avoid overly penalizing longer sentences). We also compute the overall SR as an average across all sentence lengths, the overall MSE as an average across all sentence lengths, and

the overall normalized MSE as an average over all sentence lengths. These results are reported in Table 3.

Table 3: Overall SR and MSE Evaluation

Generation setting	SR	MSE	Normalized MSE
WCC tokens only	0.613	169.383	3.085
WCC tokens and text prompt	0.779	71.466	1.888
Text prompt only	0.969	6.769	0.271

Conditioning on only WCC token The first generation setting we evaluate on is when the model is only prompted with an WCC token. Without any other restrictions, can the model generate a sentence of the desired length? Plots of the SR and MSE for this setting are in Figure 1.

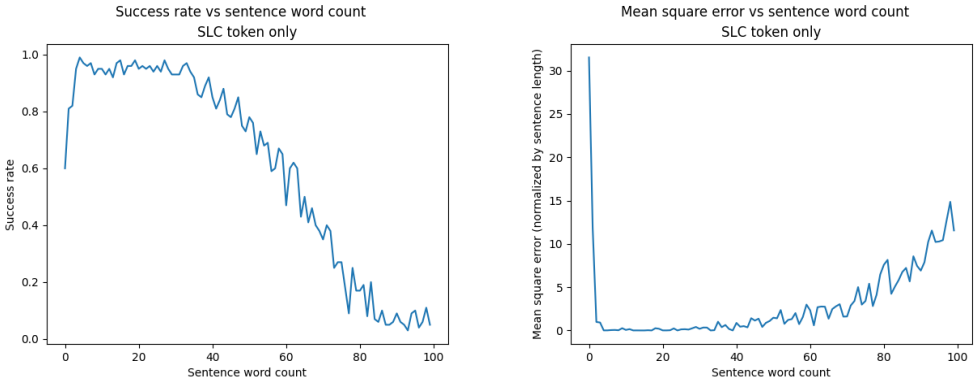

Figure 1: SR and MSE for conditioning on only WCC token

Conditioning on WCC token and text prompt The second generation setting we evaluate on is when the model is prompted with a previous sentence in English and then asked to generate a sentence of length given by the WCC token. Does the additional soft restriction of generating text that follows the previous sentence affect the model's ability to generate sentences of the desired length? Plots of the SR and MSE for this setting are in Figure 2. Examples of text generated by the model using this setting are in Appendix A.2.

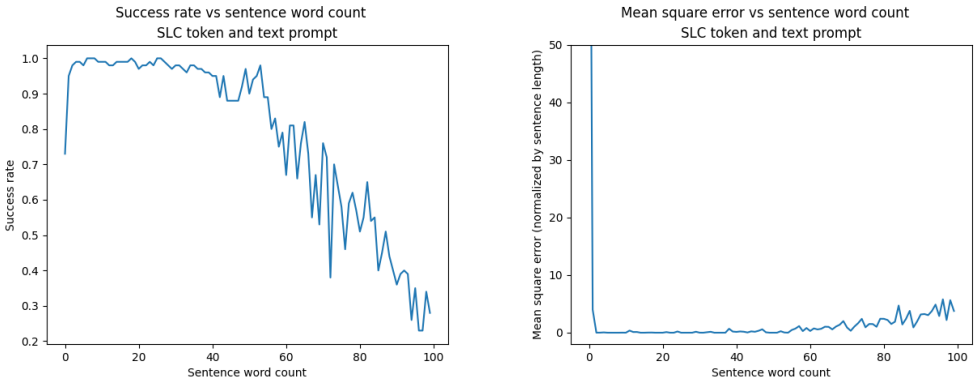

Figure 2: SR and MSE for conditioning on WCC token and text prompt

Conditioning on only text prompt The third generation setting we evaluate is when the model is prompted with a previous sentence in English but not an WCC token. As a result, the model must pick its own WCC token and then generate a sentence with the corresponding number of words. This setting is intended to evaluate the model's ability to create a plan and then follow through with that

plan. We use the same prompt sentences for this generation setting as for the previous generation setting. Plots of the SR and MSE for this setting are in Figure 3. Notice that the axis on the plots for this setting only go up to 52; the model never picked an WCC token corresponding to a sentence length higher than 52 during the evaluation trials. A histogram of the chosen sentence lengths by the model is in Figure 4.

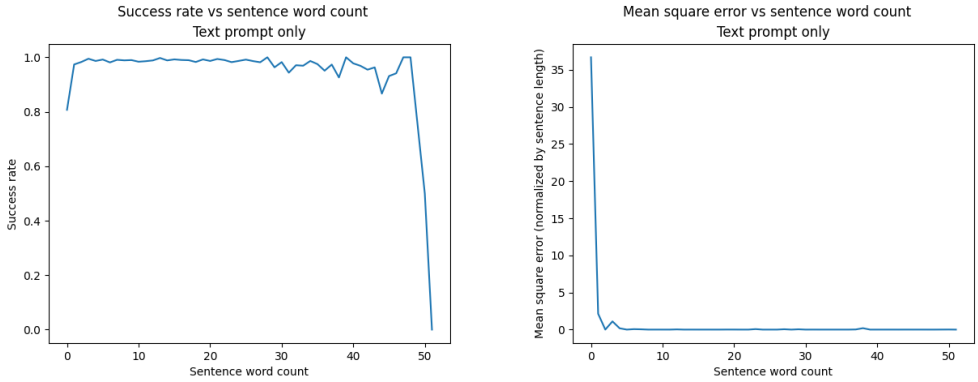

Figure 3: SR and MSE for conditioning on text prompt only

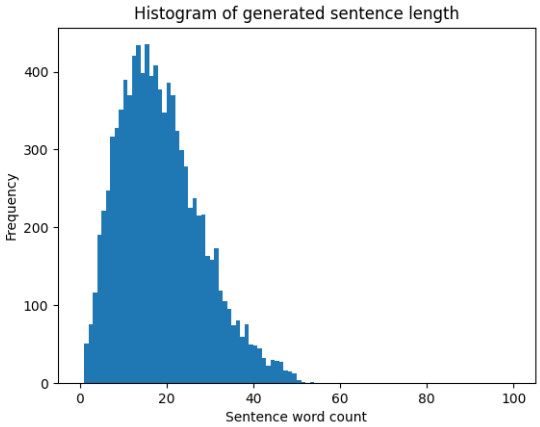

Figure 4: Histogram of chosen sentence lengths in text prompt only setting

6 Analysis

We notice several trends in the behavior of the model from the results above.

First, between the three generation settings, the SR is highest and the MSE is lowest when the model can choose its own sentence length. This makes sense, since the model can choose a sentence length that suits its purposes and the prompt sentence best instead of being forced to write a sentence of a possibly awkward length (like a very short sentence of length 1 or a very long sentence of length 100). Interestingly, the SR and MSE are higher when a text prompt is given than when there is no text prompt and the model is only prompted with the WCC token. This could be because the text prompt helps put the model in the distribution of generating text in a sentence format rather than some other format of text that may have been in the dataset but was not properly filtered out, such as code or a list.

We notice that when the model can choose its own sentence length in the conditioning on only text prompt setting, the model usually chooses a sentence length between 1 and 50 words. This makes sense, since it is known that most English sentences are less than 50 words long (Sigurd et al., 2004).

Examining the distribution of sentence lengths in a sample of OpenWebText corroborates this claim (see the histogram in Figure 5).

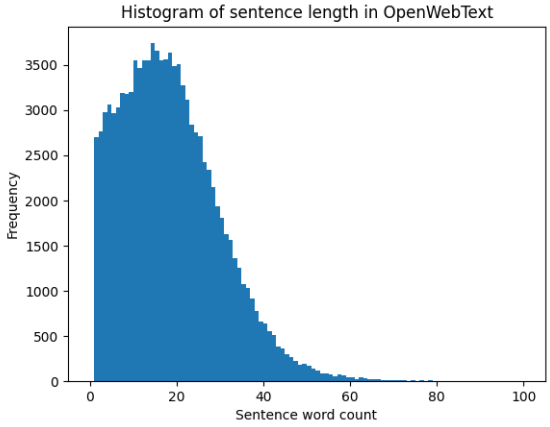

Figure 5: Histogram of sentence lengths in OpenWebText

Next, the SR decreases and MSE increases as the sentence length increases, even when the MSE is normalized by sentence length, meaning it is harder for the model to generate a longer sentence of the correct length than a shorter one. The exception to this is for sentences of length 1, since the model has a significantly higher error rate on sentences of length 1 that then sharply improves at length 2 and above. This could be partly because there are less examples in the dataset of sentences with longer length (see Figure 5), so the model is not as familiar with them. This could also be because generating a longer sentence is more difficult in general, since the model needs to remember information farther back in its context about the content and grammar of the sentence. In the specific case of 1-word sentences, we notice that a common mistake made by the model is to generate a word that ends with a period but does not make a grammatical sentence, such as `Prof.` or `P.S.`. This could be due to the model learning to associate periods with the ends of sentences and mistakenly counting a word that ends with a period as the end of a sentence.

We also note that the SR and MSE on each individual prompt sentence is fairly uniform. The SR varies between roughly 0.6 and 0.9, with no clear trend in the types of prompt sentences the model performs better or more poorly on.

7 Conclusion

Our method of achieving word count control is incredibly effective, easily outperfomring the state-of-the-art ChatGPT and finetuned GPT-2 large models despite using a model that is orders of magnitude smaller. This implies that despite the autoregressive, left-to-right nature of Transformer decoder models, they can plan ahead in generating text. Even though ChatGPT and other widely available language models currently perform poorly at generating text with length constraints, LLMs trained to understand word count through the right methods (such as our WCC method) can achieve very high accuracy at generating text with length constraints.

This work is limited by the fact that we only test our method on a small model, so though the generation quality of text generated from our model matches that of models of equivalent size, there is some room for improvement in the text generated from the model. Likewise, the model has not been aligned or instruction tuned in any way, so it is not as easily interactive and may be harder to use for general purposes compared to ChatGPT, even if it outperforms ChatGPT at generating text with length constraints.

It may be interesting to further investigate the ability of LLMs to plan ahead when generating text under certain conditions represented by control tokens and why this works; we leave this to future work. In future work, we may also want to investigate whether this method scales to larger LLMs and wherher the quality of text generation remains unaffected by using this method at larger scales (specifically, investigating pretraining or finetuning a larger LLM on the WCC method). We may

also want to investigate whether the same method for word count control works to control the token count, syllable count, sentence count, and paragraph count of generated text. Finally, a larger-scale analysis with more prompt sentences and more generation samples investigating the effect of prompt sentence on SR and MSE may be interesting to discover whether there are any trends in the ability of the model to generate sentences of the correct length based on the content or grammatical structure of the previous sentence.

Acknowledgments

We would like to thank Caleb Ziems, who this project was mentored by and who provided crucial insight and direction!

Thanks also to John Thickstun, who helped suggest using MAUVE to evaluate the quality of the generations from the model, and Tatsu Hashimoto, who helped suggest papers to read when this project was in its initial stages.

The code for training the models is derived from the GPT-2 implementation in the Levanter repository created by David Hall and Ivan Zhou for training foundation models using Jax: `https://github.com/stanford-crfm/levanter/tree/main`.

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.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Julie Kallini, Isabel Papadimitriou, Richard Futrell, Kyle Mahowald, and Christopher Potts. 2024. Mission: Impossible language models.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. 2022. Diffusion-lm improves controllable text generation. *ArXiv*, abs/2205.14217.

Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv: Learning*.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems*, volume 34, pages 4816–4828. Curran Associates, Inc.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Bengt Sigurd, Mats Eeg-Olofsson, and Joost van de Weijer. 2004. Word length, sentence length and frequency – zipf revisited. *Studia Linguistica*, 58:37 – 52.

Alexander Spangher, Yao Ming, Xinyu Hua, and Nanyun Peng. 2022. Sequentially controlled text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6848–6866, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Wieting, Nanyun Peng, and Xuezhe Ma. 2023. Evaluating large language models on controlled generation tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3155–3168, Singapore. Association for Computational Linguistics.

Yufei Tian, Anjali Narayan-Chen, Shereen Oraby, Alessandra Cervone, Gunnar Sigurdsson, Chenyang Tao, Wenbo Zhao, Yiwen Chen, Tagyoung Chung, Jing Huang, and Nanyun Peng. 2023. Unsupervised melody-to-lyric generation.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2022. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56:1 – 37.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 42602–42613. PMLR.

A Appendix

A.1 Prompt sentences for generation evaluation

We used ChatGPT to generate sentences for evaluating the WCC method conditioned on a text prompt. The prompt given to ChatGPT is as follows:

```
You are evaluating an LLM for the quality of its generated
sentences.  The LLM is prompted with one sentence and asked
to generate a second sentence to follow the previous sentence.
The second sentence should logically and grammatically follow
the first.  Generate several sentences to be the prompt
sentence for the LLM. Generate sentences that suggest a
particular topic.  An example sentence is, " Here is some
advice about how to train a dog."
```

A sample of the first 10 out of 100 text prompt sentences is below:

```
Let's explore the benefits of mindfulness meditation for stress relief.
Exploring the fascinating world of astrophysics, let's talk about black holes.
The key to a successful presentation is practice and preparation.
Here's a recipe for making fluffy pancakes from scratch.
Today, we're diving into the history of ancient civilizations.
Let's explore the impact of climate change on marine ecosystems.
Today's topic for discussion is the future of renewable energy.
The key to a healthy diet is balancing nutrients and portion control.
The first step in building a successful business is identifying your target market.
The importance of getting enough sleep cannot be overstated.
```

A.2 Text generation examples

Below are a few examples of text generated from the model on the prompts above. All of the generated samples below were prompted with a sentence length of 5 words. Note that all of the generated sentences are the correct length, since our model achieved 100% accuracy (SR 1.0) on generating sentences 5 words long.

```
Let's explore the benefits of mindfulness meditation for stress relief.
So what does this mean?
```

Exploring the fascinating world of astrophysics, let's talk about black holes.
What is an active galaxy?

The key to a successful presentation is practice and preparation.
Practice helps you gain insights.

Here's a recipe for making fluffy pancakes from scratch.
Preheat oven to 400 degrees.

Today, we're diving into the history of ancient civilizations.
That means understanding their motivations.

Let's explore the impact of climate change on marine ecosystems.
It's worth exploring marine environments.

Today's topic for discussion is the future of renewable energy.
This is not just talk.

The key to a healthy diet is balancing nutrients and portion control.
Your body needs two things.

The first step in building a successful business is identifying your target market.
A target market is anyone.

The importance of getting enough sleep cannot be overstated.
Every minute counts for more.