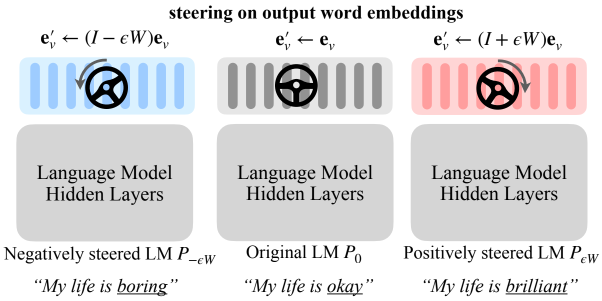
Word Embeddings Are Steers for Language Models

Chi Han, Jialiang Xu, Manling Li, Yi Fung, Chenkai Sun,
Nan Jiang, Tarek Abdelzaher, Heng Ji
University of Illinois Urbana-Champaign
{chihan3, jx17, manling2, yifung2, chenkai5
nanjiang, zaher, hengji}@illinois.edu

Abstract

Language models (LMs) automatically learn word embeddings during pre-training on language corpora. Although word embeddings are usually interpreted as feature vectors for individual words, their roles in language model generation remain underexplored. In this work, we theoretically and empirically revisit output word embeddings and find that their linear transformations are equivalent to steering language model generation styles. We name such steers LM-Steers and find them existing in LMs of all sizes. It requires learning parameters equal to 0.2% of the original LMs' size for steering each style. On tasks such as language model detoxification and sentiment control, LM-Steers can achieve comparable or superior performance compared with state-of-the-art controlled generation methods while maintaining a better balance with generation quality. The learned LM-Steer serves as a lens in text styles: it reveals that word embeddings are interpretable when associated with language model generations and can highlight text spans that most indicate the style differences. An LM-Steer is transferrable between different language models by an explicit-form calculation. One can also continuously steer LMs simply by scaling the LM-Steer or compose multiple LM-Steers by adding their transformations. Our codes are publicly available at `https://github.com/Glaciohound/LM-Steer`. [1]



steering on output word embeddings
$\mathbf{e}'_v \leftarrow (I - \epsilon W)\mathbf{e}_v$
$\mathbf{e}'_v \leftarrow \mathbf{e}_v$
$\mathbf{e}'_v \leftarrow (I + \epsilon W)\mathbf{e}_v$
Language Model Hidden Layers
Language Model Hidden Layers
Language Model Hidden Layers
Negatively steered LM $P_{-\epsilon W}$
Original LM P_0
Positively steered LM $P_{\epsilon W}$
"My life is boring"
"My life is okay"
"My life is brilliant"

Figure 1: We find hidden steers in **output word embeddings**. By linearly transforming word embeddings, language model generations are "steered" toward different style polarity and levels.

1 Introduction

In recent years, language models (LMs) have significantly advanced various natural language processing (NLP) tasks such as machine translation, sentiment analysis, schema induction, summarization, and sociocultural understanding (Brown et al., 2020; Kojima et al.; Li et al., 2023b; Radford et al., 2018; OpenAI, 2023; Fung et al., 2023, 2024). Their **output word embeddings** are learned automatically to calculate word output likelihoods during pre-training on language corpora. Typically, the dot product $\mathbf{c}^\top \mathbf{e}_v$ between a computed context vector and a learnable output word embedding $\mathbf{e}_v$ for token v is usually used as the word logit. The word output probability is defined as the softmax over all word logits:

$$P(v|\mathbf{c}) = \frac{\exp(\mathbf{c}^\top \mathbf{e}_v)}{\sum_{u \in \mathcal{V}} \exp(\mathbf{c}^\top \mathbf{e}_u)}, \qquad (1)$$

where $\mathcal{V}$ is the whole vocabulary. While being a fundamental topic in natural language processing, previous work on interpreting them is usually focused at the word level, such as their semantic information (Şenel et al., 2018), word senses (Hewitt et al., 2023), and analogical relations (Mikolov et al., 2013; Park et al., 2017). However, as the word embeddings are optimized for generation loss during pre-training, the learned embedding space should be closely associated with LMs' generation distributions. In this work, we propose to study the roles that word embeddings play in LM generation, which remains an underexplored topic, and analyze a simple while effective LM steering method LM-Steer.

[1] Please be advised that this paper contains potentially controversial results and examples to some readers, included solely for research purposes to explore model capabilities.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16410–16430
August 11-16, 2024 ©2024 Association for Computational Linguistics

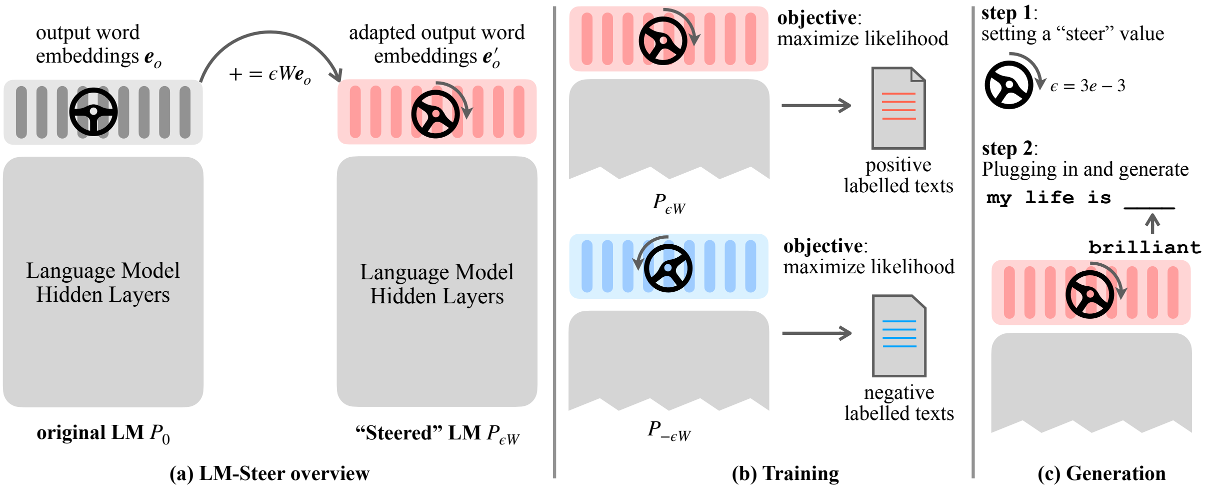

output word embeddings e_o
$+ = \epsilon W e_o$
adapted output word embeddings e'_o
Language Model Hidden Layers
Language Model Hidden Layers
original LM P_0
"Steered" LM $P_{\epsilon W}$
(a) LM-Steer overview
objective: maximize likelihood
$P_{\epsilon W}$
positive labelled texts
objective: maximize likelihood
$P_{-\epsilon W}$
negative labelled texts
(b) Training
step 1: setting a "steer" value
$\epsilon = 3e-3$
step 2: Plugging in and generate
my life is ____
brilliant
(c) Generation

Figure 2: An overview of LM-Steer. **(a)**: LM-Steer applies a linear factor $\epsilon W \mathbf{e}_v$ to each word embedding for language model conditioning. **(b)**: During training, we use a positively steered model $P_{\epsilon W}$ to maximize likelihood on positively labeled texts, and vice versa. **(c)**: For generation, one only needs to specify a steering value ϵ, and then proceed with normal decoding.

We start by examining the theoretical relation between word embeddings and LM generation distributions. We find that linear transformations in output word embeddings are equivalent to LM generation style changes. This motivates our empirical investigation on a lightweight and simple steering method, LM-Steer, to steer LM generation distribution flexibly and transparently. LM-Steer deploys a $d \times d$ learnable linear transformation W on the output word embeddings, where d is the embedding dimension. Specifically, the embeddings $\mathbf{e}_v$ are replaced with $\mathbf{e}_v + \epsilon W \mathbf{e}_v$. Here, ϵ acts as a "steering value" to provide a simple control on steering polarity and intensity. Inherently, LM-Steer discovers hidden dimensions in word embeddings that are associated with text styles. By transforming these dimensions, LM-Steer affects LMs' interpretation of words and, consequently, generation distributions.

Empirically, we find such LM-Steers exist prevalently in LMs thanks to its general formulation and ignorance of model architectures. It can efficiently steer LMs of all sizes, ranging from smaller GPT-2 family (Radford et al., 2019) to modern large language models, including GPT-J (Wang and Komatsuzaki, 2021) and Llama-2-7B (Touvron et al., 2023). On tasks such as language model detoxification and sentiment control, it achieves comparable or superior performance compared to the state-of-the-art controlled generation baselines. We also find multiple merits of LM-Steer both as a steering tool and lens for inspecting the relation between word embeddings and language model generation. An LM-Steer can **highlight** text spans that best indicate a style in the full text. A learned LM-Steer also makes word embedding dimensions **interpretable** by pointing out the dimensions closest related to a style, revealing what kinds of words contribute to or contradict a style. LM-Steer is both **parameter efficient** and **data-efficient**: on GPT2-large, it learns only parameters only 0.2% the size of the original model (9% of the size of LoRA (Hu et al., 2021a), a parameter-efficient fine-tuning method) and able to text sentiments on dozens of sentences. A learned LM-Steer is **transferable** to other LMs with different sizes by explicit-form calculation without additional training. Moreover, LM-Steer theoretically enables both **continuous** and **compositional** control. This allows for dealing with diverse and nuanced situations, such as fine-grained personalized or customized generation, without re-training for each scenario.

2 Related Work

Understanding Word Embeddings Language models learn word embeddings for individual words automatically after pre-training. To understand them, (Mikolov et al., 2013; Allen and Hospedales, 2019) discover linear translational relations among embeddings, while (Park et al., 2017; Rothe and Schütze, 2016; Ethayarajh, 2019) examine their rotational relations. Some other work examines the semantic information (Şenel et al., 2018; Lund and Burgess, 1996; Jang and Myaeng, 2017; Şenel et al., 2022; Murphy et al., 2012; Faruqui et al., 2014) or word senses (Panigrahi et al., 2019; Hewitt et al., 2023) of individual words in their embeddings. These efforts, however, have mostly focused on and evaluated word-level interpretations of word embeddings (Chang et al., 2009) while we first investigate their relations with language model generations.

Control of Language Models has been of growing interest in recent years, motivated by the increasing capabilities of LMs (Li et al., 2023a). This area originates from the need to leverage the generation capabilities of large language models while avoiding the need for time-consuming and costly retraining or fine-tuning. Attempts include applying attribute classifiers or heuristic constraints at decoding time (Kumar et al., 2022; Dathathri et al.; Liu et al., 2021; Yang and Klein, 2021), treating the generation process as an optimization problem over the embedding or token sequences (Kumar et al., 2021), or post-editing the output (Li et al., 2018). These techniques are often computationally expensive and rely on suitable external classifiers. More recently, prompting-based control for large language models has received much attention, which, however, relies on the quality and availability of large language models (Brown et al., 2020; OpenAI, 2023), and may also necessitate the deliberate training (Raffel et al., 2020; Zhou et al., 2023). It can also be challenging to design effective prompts for complex or nuanced control goals. Parameter-efficient fine-tuning such as LoRA (Hu et al., 2021a) focuses on learning low-rank approximations of model parameters. However, this cannot achieve flexible and transferrable language model steering like ours. Probably most closely related to our work are attempts to discover "steering" vectors or tokens (Subramani et al., 2022; Li and Liang, 2021), and also similar work in image generation (Jahanian et al.; Hu et al., 2021b). Different from our model, these efforts focus on other applications such as multi-task learning and sentence recovery, and the learned vectors are not shown to be transferrable or interpretable nor enable flexible control.

Language Model Detoxification Motivated by the goal to address the systematic biases embedded in language models, there are efforts in conducting language model de-biasing or de-toxification (Meade et al., 2022; Kaneko et al., 2022). Approaches span all aspects of the language model pipeline. A line of work focuses on automatically obtaining cleaner data (Barikeri et al., 2021; Webster et al., 2021; Dinan et al., 2020). Another line of work modifies the model workflow design to explicitly accommodate the bias factors (Webster et al., 2021; Schick et al., 2021; Yu et al., 2023; Omrani et al., 2023; Yang et al., 2023). The most related line of work to the herein proposed method involves manipulating embedding space such as Principle Component Analysis and Nullspace Projection (Liang et al., 2020; Bolukbasi et al., 2016; Ravfogel et al., 2020). The evaluation in these settings (Kaneko and Bollegala, 2021; Nadeem et al., 2021; Nangia et al., 2020) mostly consists of quiz-question checking for stereotypical misbeliefs. More related to our method are those mentioned in language model control (Kumar et al., 2022; Dathathri et al.; Liu et al., 2021; Yang and Klein, 2021; Kumar et al., 2021), which constrains or guides text generation according to a classifier. A unique contribution in our work is that the learned LM-Steer can be transferred to detoxify other off-the-shelf language models without a costly training process.

3 LM-Steer: Revealing Hidden Steers in Word Embeddings

As a theoretical motivation, we first show an informal theorem relating output word embeddings with generation styles. We leave the formal statement as well as the proof to Appendix C and only present an intuitive interpretation.

Theorem 1. *(Informal) With certain assumptions, shifting styles in language models is equivalent to a linear transformation in word embedding space.*

Inspired by this discovery, we propose LM-Steer to apply a linear transform in the output word embedding space. LM-Steer is conceptually simple and straightforward to implement. An illustration of LM-Steer is presented in Figure 2(a). Specifically, we assume a language model with fixed parameters. We replace its each output word embeddings $\mathbf{e}_v$ with

$$\mathbf{e}'_v = \mathbf{e}_v + \epsilon W \mathbf{e}_v = (I + \epsilon W)\mathbf{e}_v, \qquad (2)$$

and call the resulting language model $P_{\epsilon W}$ a "LM-Steered model". Here, the "steer matrix" W is the only learnable parameter in LM-Steer, and ϵ is an adjustable scalar indicating the polarity and intensity of the "steering value". Without loss of generality, we arbitrarily pick a small value $\epsilon_0 = 1e - 3$ as the default steering value.[2] We use $P_{\epsilon W}$ to

[2] Using an arbitrary ϵ_0 possess the same representation

denote the steered language model. Figure 2(b, c) shows the training and generation process of LM-Steer. During training, we use the positively steered model $P_{\epsilon W}$ to fit the positively labeled texts, with maximal likelihood as the training objective. When negative texts are available, we also fit them with $P_{-\epsilon W}$. When generating with LM-Steer, the user only needs to specify a steering value ϵ and then decode the language model. More details are in Appendix D. Intuitively explaining, LM-Steer matrix W learns to identify word embedding dimensions that are best associated with a target style and manipulate among those dimensions to achieve language rewording. In Section 5.1, we show this enables an interpretation of word embeddings by analyzing the learned W.

We also theoretically and empirically compare LM-Steer against a simplified version, a **(soft) word blacklist (SWB)**: learning a global logit offset is applied to each token candidate after the original logits are computed. As we demonstrate in Appendix I, adding a (learnable) vector to context vectors $\boldsymbol{c}$ achieves a similar effect with SWB. We also further prove that LM-Steer is theoretically expressive of any distribution shift, while a SWB is unable to do so. In 4, we show that SWB indeed yields inferior performance than LM-Steer.

4 Steering Language Model Generation

4.1 Language Detoxification

It is known that large pre-trained LMs might generate toxic content that appears in the pre-training distribution (Sheng et al., 2019; Gehman et al., 2020), such as inaccurate information, harmful stereotypes, and unethical content. Language model detoxification is the task of mitigating or avoiding these generations in order to enable the safe usage of language models. We experiment on the GPT2 family, Pythia family (Biderman et al., 2023), GPT-J-6B, and Llama-2-7B as the backbone language models.

Setting: Following (Liu et al., 2021), we use Jigsaw Unintended Bias in Toxicity Classification Kaggle challenge[3] as the training dataset. For evaluation, we use 10K nontoxic prompts from the RealToxicityPrompts dataset (Gehman et al., 2020). We randomly generate 25 sentences of up to 20 tokens using nucleus sampling (Holtzman et al.) with $p = 0.9$. Then the toxicity scores (in range $[0, 1]$) of generations are evaluated using Perspective API [4]. Two metrics are reported: the maximal toxicity of generations on each prompt averaged across prompts ("Avg. max. toxicity") and the averaged probability of generating > 0.5 toxicity ("Toxicity prob."). We also evaluate generation quality in terms of fluency (perplexity score measured by a GPT2-large) and diversity (Dist-$\{1, 2, 3\}$: the portion of distinct $\{1, 2, 3\}$-grams). When decoding, we use a steering value of $5\epsilon_0$ for generation, selected based on the balance between generation fluency and task performance on the dev set in Appendix E.

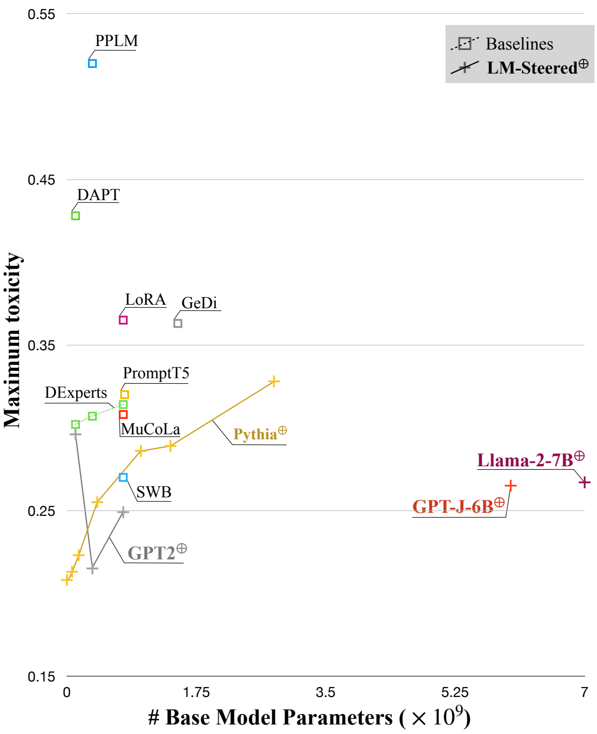


Figure 3: Across base model sizes, LM-Steered GPT2 family, Pythia family, GPT-J and Llama-2-7B models (+) consistently outperform other baselines ((□)) on detoxification. X$^{\oplus}$ means an LM-Steered language model X.

Baselines: DExperts (Liu et al., 2021) trains positive and negative label classifiers and uses the difference in two classifiers' scores to offset the LM's original logits. **DAPT** (Gururangan et al., 2020) simply further pretrains the language model on the non-toxic subset (filtered by Perspective API) of OpenWebText Corpus (OWT) (Gokaslan et al., 2019). **PPLM** (Dathathri et al.) learns to use the gradients of the label classifier to update the LM's hidden representations. **GeDi** (Krause et al.,

power as any other ϵ, as there always exists $W' = \frac{\epsilon}{\epsilon_0} W$ so that $\epsilon_0 W' = \epsilon W$ holds.

[3] https://bit.ly/3cvG5py

[4] https://perspectiveapi.com

Model	Backbone	Toxicity↓		Fluency	Diversity↑		
	Size	Max. toxicity	Toxicity prob.	Output ppl.↓	Dist-1	Dist-2	Dist-3
GPT-2 (original)	117M	0.527	0.520	25.45	0.58	0.85	0.85
PPLM (10%)	345M	0.520	0.518	32.58	0.58	0.86	0.86
DAPT	117M	0.428	0.360	31.21	0.57	0.84	0.84
GeDi	1.5B	0.363	0.217	60.03	0.62	0.84	0.83
DExperts$_{\text{base}}$	117M	0.302	0.118	38.20	0.56	0.82	0.83
DExperts$_{\text{medium}}$	345M	0.307	0.125	32.51	0.57	0.84	0.84
DExperts$_{\text{large}}$	762M	0.314	0.128	32.41	0.58	0.84	0.84
PromptT5	780M	0.320	0.172	55.1	0.58	0.76	0.70
MuCoLa	762M	0.308	0.088	29.92	0.55	0.82	0.83
LoRA	762M	0.365	0.210	21.11	0.53	0.85	0.86
Soft-Blacklist	762M	0.270	0.154	18.28	0.53	0.81	0.83
LM-Steer$_{\text{base}}$	117M	$0.296_{\pm 0.018}$	$0.129_{\pm 0.012}$	36.87	0.54	0.86	0.86
LM-Steer$_{\text{medium}}$	345M	$\mathbf{0.215}_{\pm 0.015}$	$\mathbf{0.059}_{\pm 0.029}$	43.56	0.56	0.83	0.84
LM-Steer$_{\text{large}}$	762M	$0.249_{\pm 0.007}$	$0.089_{\pm 0.009}$	28.26	0.55	0.84	0.84

Table 1: On language model detoxification, LM-Steer achieves best performance. ± denotes standard deviation on 3 random seeds.

	LM-Steer	Tie	LoRA	LM-Steer	Tie	GPT-2	LM-Steer	Tie	DExperts
Detoxified	**19.0**	69.5	11.5	**24.5**	56.5	19.0	**24.0**	56.5	19.5
Fluent	**21.0**	69.0	10.0	21.0	57.5	**21.5**	**25.0**	52.0	23.0
Topical	**18.0**	69.5	12.5	**32.0**	47.0	21.0	**32.0**	56.5	11.5

Table 2: Human evaluation results by comparing with LoRA, GPT-2 and DExperts. LM-Steer wins out on most metrics while being comparable to GPT-2 on fluency.

2021) is a model that uses the Bayesian rule for class-conditioned LM generation. **MuCoLa** (Kumar et al., 2022) models the text generation as an optimization problem regarding the classifier scores. **PromptT5** (Raffel et al., 2020) T5 is a pre-trained LM optimized for prompt-based task solving, and we use "Complete this sentence so that it embodies a {positive/negative} sentiment:" to prompt T5. **LoRA** (Hu et al., 2021a) trains low-rank approximations of parameter matrices to achieve parameter-efficient fine-tuning. Finally, we compare with the soft blacklist baseline discussed in Section 3.

Results and Analysis: We present the results in Table 1. Despite the simple design, LM-Steer achieves the best detoxification scores on both metrics, reducing Avg. max. toxicity by $>6\%$ absolute percentages. It is also noteworthy that LM-Steer also demonstrates reasonable balance on fluency (2nd lowest perplexity score) and diversity (same-level Dist-k scores with baselines). Figure 3 further shows the detoxification versus baseline size, where LM-Steer+{GPT2 family, Pythia family, GPT-J and Llama-2} uniformly outperforms baselines where of all sizes, where more numerical results can be found in Appendix J and M. Incorporation of LM-Steer with LoRA, instruction following, and full embedding tuning are explored in Appendix L, N and O, respectively.

Human Evaluation We compare LM-steer with LoRA, DExperts, and GPT-2 in a pairwise manner with human annotators. Specifically, we follow the practice in DExperts and ask four student human annotators to compare 50 generations from LM-steer and the baseline from 3 perspectives: detoxification, fluency, and being topical to the prompt. The results are as follows. We can see that LM-steer is ranked significantly less toxic and more topical than the baseline. It performs similarly to DExperts and GPT-2 but better than LoRA in terms of fluency.

4.2 Sentiment Control

We also evaluate LM-Steer's performance on an extensively studied generation task controlled by sentiment. This ability can be found useful when tailor-

Target	Model	Sentiment Positivity / %			Fluency	Diversity↑		
		Positive prompts	Neutral prompts	Negative prompts	Output ppl.↓	Dist-1	Dist-2	Dist-3
Positive↑	LM-Steer$_{\text{large}}$		90.70	41.23	41.20	0.46	0.78	0.83
	LM-Steer$_{\text{medium}}$		**95.36**	56.98	67.68	0.46	0.77	0.80
	LM-Steer$_{\text{base}}$		90.46	**57.26**	54.38	0.47	0.78	0.81
	Soft-Blacklist		86.40	25.64	99.46	0.42	0.76	0.81
	LoRA		26.88	7.20	158.56	0.57	0.82	0.83
	DExperts$_{\text{large}}$		94.46	36.42	45.83	0.56	0.83	0.83
	DExperts$_{\text{medium}}$		94.31	33.20	43.19	0.56	0.83	0.83
	DExperts$_{\text{small}}$		94.57	31.64	42.08	0.56	0.83	0.84
	DExperts (pos)		79.83	43.80	64.32	0.59	0.86	0.85
	GeDi		86.01	26.80	58.41	0.57	0.80	0.79
	DAPT		77.24	14.17	30.52	0.56	0.83	0.84
	PPLM (10%)		52.68	8.72	142.11	0.62	0.86	0.85
	PromptT5		68.12	15.41	37.3	0.58	0.78	0.72
	GPT-2 (original)	99.08	50.02	0.00	29.28	0.58	0.84	0.84
Negative↓	PromptT5	69.93	25.78		48.6	0.60	0.78	0.70
	PPLM (10%)	89.74	39.05		181.78	0.63	0.87	0.86
	DAPT	87.43	33.28		32.86	0.58	0.85	0.84
	GeDi	39.57	8.73		84.11	0.63	0.84	0.82
	DExperts (neg)	61.67	24.32		65.11	0.60	0.86	0.85
	DExperts$_{\text{small}}$	45.25	3.85		39.92	0.59	0.85	0.84
	DExperts$_{\text{medium}}$	40.21	3.79		43.47	0.59	0.85	0.84
	DExperts$_{\text{large}}$	**35.99**	**3.77**		45.91	0.60	0.84	0.83
	LoRA	57.71	20.08		192.13	0.55	0.78	0.79
	Soft-Blacklist	73.72	14.28		50.95	0.38	0.70	0.76
	LM-Steer$_{\text{base}}$	57.26	10.12		51.37	0.49	0.77	0.79
	LM-Steer$_{\text{medium}}$	52.32	7.10		71.48	0.47	0.77	0.79
	LM-Steer$_{\text{large}}$	54.84	8.02		57.74	0.48	0.78	0.80

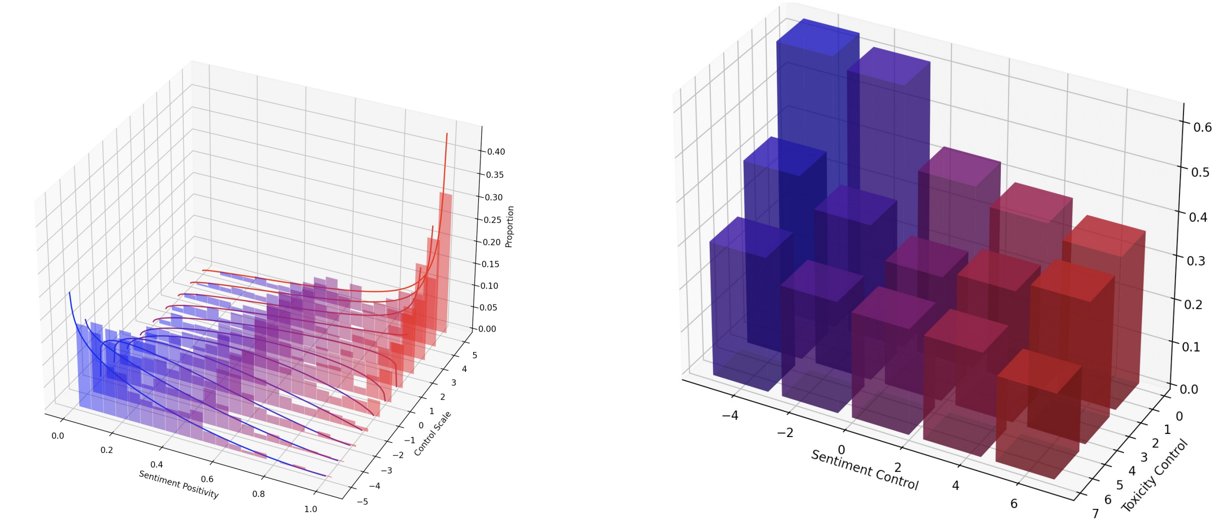
Table 3: Results on sentiment control task. The upper half displays a positive control task and requires a higher positivity score and vice versa for the lower half. LM-Steer gets the best metrics on the positive side and 2nd to 3rd places on the negative side despite being simpler and smaller. For backbone model sizes, please refer to Table 1.

ing persuasive and emotionally appealing messages to specific target audiences in marketing or advertising or to create personalized and engaging user experiences in chatbot systems.

Setting: We follow the setting in (Liu et al., 2021) and use Stanford Sentiment Treebank (SST-5) (Socher et al., 2013) as training data, where we use texts with labels $1\sim2$ as negative samples, and those with $4\sim5$ labels as positive samples. For evaluation, we use the HuggingFace's sentiment classifier (Wolf et al., 2020). The generation prompts are a subset of the OpenWebText Corpus filtered by the sentiment analysis classifier. Models are applied on these prompts 25 times to generate up to 20 tokens. We then measure the average percentage of positive generations for each prompt as the "Positivity" score. Similar to the detoxification task, we use $5\epsilon_0$ for positive sentiment and $-5\epsilon_0$ for negative sentiment control.

Baselines: In addition to the baselines used in detoxification, two variants of DExperts, DExperts (pos) and DExperts (neg), which only use one of the two classifiers for guiding generation, are also listed.

Results: Table 3 presents the full results. LM-Steer, despite a much simpler and smaller model, takes 1st place on the positive side and 2nd or 3rd place on the negative side and achieves a reasonable balance on fluency and diversity.



(a) Continuous control on sentiment with ϵ in $-5\epsilon_0 \sim 5\epsilon_0$ results in a sentiment distribution shift. Color indicates sentiment and height indicates frequency/density.

(b) Compositional control sentiment ranging in $-5\epsilon_0 \sim 5\epsilon_0$ and toxicity in $0 \sim 5\epsilon_0$. Color means sentiment and height is toxicity.

Figure 4: Continuous and compositional control using LM-Steer.

	LM-Steer	**DAPT**	**GeDi**	**CTRL**	**PPLM**	**DExpert**	**MuCoLa**	**LoRA**
Parameters	**1.6M**	355M	355M	355M	124M	355M	898M	18M
Speed Ratio	1.24	**1.00**	2.94	3.79	270.11	1.98	24.03	**1.00**

Table 4: Decoding time and learnable parameter efficiency. Time efficiency is measured by relative decoding time compared to the base language model. The best numbers are bolded.

4.3 Continuous and Compositional Control

The conceptually simple design of LM-Steer makes it an architecture-agnostic plug-in to diverse language models. We demonstrate that LM-Steer maintains a linearity guarantee, which enables continuous and compositional control. More specifically, our model allows for interpolation and extrapolation on the steering spectrum by simply interpolating and extrapolating the steering value. Moreover, if two LM-Steers $\epsilon_1 W_1, \epsilon_2 W_2$ are learned on potentially different tasks, their effect can be combined by decoding with $P_{\epsilon_1 W_1 + \epsilon_2 W_2}$.

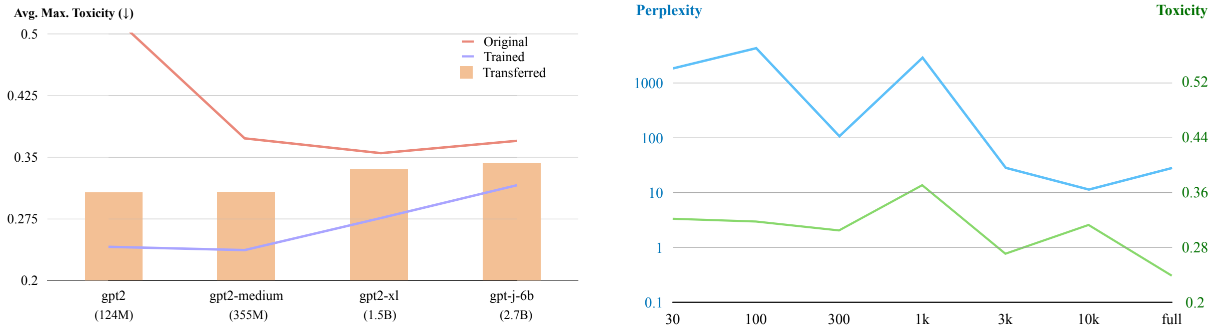
In Figure 4a, we plot the distribution shift when adjusting sentiment steer ϵ. We also curve the maximal likelihood estimated Beta distribution. In Figure 4b, we observe that LM-Steer can compositionally control sentiment and toxicity, even though there exists a mutual influence between these two factors (e.g., a negative sentiment might also lead to more toxic comments). Table 5 also provides an example of how the generated sentence is continuously steered from toxic to non-toxic, demonstrating a simple fine-grained control on the toxicity level. When the steering value increases from negative to positive, both the number and the intensity of toxic words (bolded in the table) decrease.

4.4 Efficiency

Thanks to its simple design, LM-Steer enjoys efficiency in multiple perspectives. We vary the detoxification dataset size from 30 to 10k and measure LM-Steer's performance in Figure 5(b). We see that as few as 30 data points still enable LM-Steer to achieve high detoxification scores (0.322). When dataset size exceeds 3k LM-Steer acquires a good balance between detoxification and generation quality. We also show decoding time and parameter efficiency in Table 4, where our model only uses 1% of the baseline's learnable parameter size and uses a low computation overhead during decoding.

5 LM-Steers Connect Word Embeddings with the Text Distribution

In previous sections, LM-Steer revealed the hidden biases encoded in the automatically learned word embeddings of LMs. This section provides an alternative perspective, where these hidden biases serve as a lens for interpreting the connection between word embeddings and the generation distribution of LMs. Section 5.1 demonstrates how LM-Steer



(a) Transferring an LM-Steer to other LMs with explicit-form calculation. The transferred LM-Steer maintains the detoxification ability partially.

(b) LM-Steer is capable of learning from only dozens of data samples to achieve decent detoxification. More data are beneficial for fluency.

Figure 5: Measuring the transferability and data efficiency of LM-Steer.

Steer	Generation
-5e-3	What **moron** said that **stupid** comment.
-3e-3	What's **stupid** is **stupid**, right?
-1e-3	What's this? You think that your religion, your culture, your country are **not good enough**?
0	What's more, it makes for a fun, cheap, and efficient way to improve the performance of your car engine and to make your driving that much safer.
1e-3	What's more, it makes for a fun, cheap, and efficient way to improve the performance of your car engine and motor.
3e-3	What's on your mind? What's on your mind?
5e-3	What's on Netflix? If you can't figure out what's being watched on Netflix, you need to figure out what are people watching!

Table 5: LM-Steer continuously steers GPT2-Large generation from toxic to non-toxic when interpolating and extrapolating the steering value. Both the number and intensity of **toxic words** decrease with increased steering value.

decomposes word embeddings into interpretable dimensions closely related to word selection in a particular distribution. Section 5.2 shows that LM-Steer can highlight indicative words. Finally, in light of the hypothesis that different LMs have correlated word embedding spaces, Section 5.3 illustrates how LM-Steer can be transferred between LMs with an explicit expression.

5.1 Interpreting Word Embeddings

LM-Steer provides a lens on how word embeddings correlate with LM word embeddings: *what word dimensions contribute to or contrast to a specific style.* In the detoxification experiment, we conduct an SVD decomposition of the learned W. Among S, V, D, the D component can be interpreted as a ranked list of the most "magnified" row dimensions in the transformation W. We then take its first 9 rows and list the most influenced words in Table 6. Dimensions 2, 4, and 6 are filtered out as they only match non-English tokens. Although offensive to read, this table helps us understand what kind of words are most related to toxicity and thus suppressed by LM-Steer in a generation. More details are explained in Appendix G.

5.2 Highlighting Keywords in Styled Texts

LM-Steer also enables automatically pointing out *what specific words are most indicative of the style in a given sentence.* We conjecture that these distinctive words contribute most to the likelihood difference between a task-specific LM and a domain-general LM. In Table 7, we list a few toxic sentences and the most indicative text spans highlighted by LM-Steer. To acquire such text spans, we calculate $\log P_{\epsilon W}(v_i|\cdots v_{i-1}) - \log P_0(v_i|\cdots v_{i-1})$ which is the difference of log-likelihoods of each token under steered ($P_{\epsilon W}$) and the original language model (P_0). Then, we deploy a dynamic programming algorithm to obtain the continuous sub-sequence under length 5 with the highest cumulative difference. We list highlighted words in toxic prompts in the RealToxicityPrompts dataset in Table 7, which shows insulting, cursing, controversial, and sexually explicit words in each sentence.

Dim.	Matched Words
0	mor, bigot, Stupid, retarded, coward, stupid, loser, clown, dumb, Dumb, losers, stupidity, garbage , idiots, fools, idiot, lame
1	stupid, idiot, Stupid, idiots, jerk, pathetic, suck, buff, stupidity, mor, damn, ignorant, fools, dumb , disgusting , damned, narcissistic, troll
3	idiot, godd, damn,
5	Balk, lur, looms, hides, shadows, Whites, slippery, winds
7	bullshit, fiat, shit, lies, injust, manipulation
8	disabled, inactive, whip, emo, partisan, spew, bombed, disconnected, gun, failing, Republicans , defeated, Jeb, blowing , bombard, ineffective, reload, destructive, flo, blown
9	winners, upside

Table 6: Word embedding dimensions that are most influenced by LM-Steer on detoxification task.

There's another controversial Hollywood racial decision that Stacey Dash is sinking her teeth into.
The UFC champ then suggested Justino is a longtime PED user with her most d**ning comments.
But I really have a question for you: Why would I go on a game show and play into the bulls**t allowing myself to be ranked by some fake competition?
I think sexism prevents this from being a real win for fat people.
If they want to be fair and non hypocritical idiots they should.

Table 7: Toxic sentences with toxic keywords highlighted by LM-Steer after training detoxification on GPT2-Large.

5.3 Transfering LM-Steer Between Models

A much-desired property of LM-Steer, because of its theoretical soundness, is its transferability to other language models. Details and derivations of LM-Steer transfer are in Appendix F. Intuitively speaking, the original logit $\mathbf{c}^\top \mathbf{e}_v$ can be understood as a similarity or matching metric between context vector $\mathbf{c}$ and word embedding $\mathbf{e}_v$. In LM-Steer, the logit is offset by ϵ times $\mathbf{c}^\top W \mathbf{e}_v$, which is also a bilinear similarity. To transform this LM-Steer to another language model, we need to map the context vectors and word embeddings between word embedding spaces $\mathbf{e}_v = H\mathbf{e}'_v$

$$\mathbf{c}^\top W \mathbf{e}_v = (H\mathbf{c}')^\top W (H\mathbf{e}'_v) = \mathbf{c}'^\top (H^\top W H)\mathbf{e}'_v \tag{3}$$

We work by first identifying a linear mapping H from target LM word embeddings to source LM word embeddings. Then, the matrix $H^\top W H$ can be inserted into the target LM as LM-Steer. This is motivated by prior work on the linear mapping between word embeddings from different models (Li et al., 2021). Finally, the calculated steering matrix is directly applied to the target LM. Figure 5(a) shows the performance after we transfer the LM-Steer learned on GPT2-large to LMs of other sizes, ranging from gpt2 (124M) to GPT-J-6B (6B). We can see a uniform improvement in transferred LM-Steers, with GPT2 and GPT2-medium getting similar scores (0.307 and 0.308) to the best baseline (DExperts).

6 Conclusions

In this work, we discover the prevalent phenomenon of word embeddings containing steers for language model generation. We demonstrate the promise and efficacy of LM-Steer, a theoretically grounded, simple, and lightweight approach for the steering of generative language models. LM-Steer can model various styles and achieve comparable or superior performance to baselines in language model detoxification and generation control. LM-Steer also allows for continuous and compositional control and can be transferred to other language models. More importantly, it provides an interpretation of how word embeddings interplay with language model generation. So far, we have only studied output word embeddings, so it is intriguing to ask whether similar phenomena apply to other components, such as input word embeddings and hidden layers.

Limitations

One limitation of LM-Steer is that it works on word embeddings and focuses on conditions related to wording. This restricts its capability to deal with more complex tasks, such as syntactic trees or persuasive techniques that involve logical reasoning. Additionally, our model is dependent on word embeddings, so the model cannot work with language model APIs that do not provide direct access to these embeddings.

Acknowledgement

This research is partially supported by U.S. DARPA KAIROS Program No. FA8750-19-2-1004, DARPA INCAS Program No. HR001121C0165, U.S. DARPA SemaFor Program No. HR001120C0123, DARPA MIPs Program No. HR00112290105, and DARPA ITM Program No. FA8650-23-C-7316. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes, notwithstanding any copyright annotation therein.

References

Carl Allen and Timothy Hospedales. 2019. Analogies explained: Towards understanding word embeddings. In *International Conference on Machine Learning*, pages 223–231. PMLR.

Soumya Barikeri, Anne Lauscher, Ivan Vulić, and Goran Glavaš. 2021. RedditBias: A real-world resource for bias evaluation and debiasing of conversational language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1941–1955, Online. Association for Computational Linguistics.

Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41(1):164–171.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan Boyd-Graber, and David Blei. 2009. Reading tea leaves: How humans interpret topic models. *Advances in neural information processing systems*, 22.

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

Emily Dinan, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. 2020. Queens are powerful too: Mitigating gender bias in dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8173–8188, Online. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. Rotate king to get queen: Word relationships as orthogonal transformations in embedding space. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3503–3508.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Yi Fung, Tuhin Chakrabarty, Hao Guo, Owen Rambow, Smaranda Muresan, and Heng Ji. 2023. NORMSAGE: Multi-lingual multi-cultural norm discovery from conversations on-the-fly. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15217–15230, Singapore. Association for Computational Linguistics.

Yi Fung, Ruining Zhao, Jae Doo, Chenkai Sun, and Heng Ji. 2024. Massively multi-cultural knowledge acquisition & lm benchmarking.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. 2019. Openwebtext corpus (2019). *URL http://Skylion007. github. io/OpenWebTextCorpus*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

John Hewitt, John Thickstun, Christopher Manning, and Percy Liang. 2023. Backpack language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9103–9125, Toronto, Canada. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021a. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Xiaodan Hu, Pengfei Yu, Kevin Knight, Heng Ji, Bo Li, and Honghui Shi. 2021b. Muse: Textual attributes guided portrait painting generation. In *Prof. The 3rd IEEE Workshop on Artificial Intelligence for Art Creation*.

Ali Jahanian, Lucy Chai, and Phillip Isola. On the" steerability" of generative adversarial networks. In *International Conference on Learning Representations*.

Kyoung-Rok Jang and Sung-Hyon Myaeng. 2017. Elucidating conceptual properties from word embeddings. In *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, pages 91–95.

Masahiro Kaneko and Danushka Bollegala. 2021. Debiasing pre-trained contextualised embeddings. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1256–1266, Online. Association for Computational Linguistics.

Masahiro Kaneko, Danushka Bollegala, and Naoaki Okazaki. 2022. Debiasing isn't enough! – on the effectiveness of debiasing MLMs and their social biases in downstream tasks. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1299–1310, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

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

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952.

Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. Controlled text generation as continuous optimization with multiple constraints. *Advances in Neural Information Processing Systems*, 34:14542–14554.

Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. 2022. Gradient-based constrained sampling from language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2251–2277.

Ariel N Lee, Cole J Hunter, and Nataniel Ruiz. 2023. Platypus: Quick, cheap, and powerful refinement of llms. *arXiv preprint arXiv:2308.07317*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874.

Sha Li, Chi Han, Pengfei Yu, Carl Edwards, Manling Li, Xingyao Wang, Yi Fung, Charles Yu, Joel Tetreault, Eduard Hovy, and Heng Ji. 2023a. Defining a new NLP playground. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11932–11951, Singapore. Association for Computational Linguistics.

Sha Li, Ruining Zhao, Manling Li, Heng Ji, Chris Callison-Burch, and Jiawei Han. 2023b. Open-domain hierarchical event schema induction by incremental prompting and verification. In *Proc. The 61st Annual Meeting of the Association for Computational Linguistics (ACL2023)*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Yuling Li, Kui Yu, and Yuhong Zhang. 2021. Learning cross-lingual mappings in imperfectly isomorphic embedding spaces. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2630–2642.

Paul Pu Liang, Irene Mengze Li, Emily Zheng, Yao Chong Lim, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Towards debiasing sentence representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5502–5515, Online. Association for Computational Linguistics.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706.

Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical cooccurrence. *Behavior research methods, instruments, & computers*, 28(2):203–208.

Nicholas Meade, Elinor Poole-Dayan, and Siva Reddy. 2022. An Empirical Survey of the Effectiveness of Debiasing Techniques for Pre-trained Language Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1878–1898, Dublin, Ireland. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*, pages 1933–1950.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

Ali Omrani, Alireza Salkhordeh Ziabari, Charles Yu, Preni Golazizian, Brendan Kennedy, Mohammad Atari, Heng Ji, and Morteza Dehghani. 2023. Social-group-agnostic bias mitigation via the stereotype content model. In *Proc. The 61st Annual Meeting of the Association for Computational Linguistics (ACL2023)*.

OpenAI. 2023. Gpt-4 technical report.

Abhishek Panigrahi, Harsha Vardhan Simhadri, and Chiranjib Bhattacharyya. 2019. Word2Sense: Sparse interpretable word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5692–5705, Florence, Italy. Association for Computational Linguistics.

Sungjoon Park, JinYeong Bak, and Alice Oh. 2017. Rotated word vector representations and their interpretability. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 401–411, Copenhagen, Denmark. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspaces. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 512–517.

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-Diagnosis and Self-Debiasing: A Proposal for Reducing Corpus-Based Bias in NLP. *Transactions of the Association for Computational Linguistics*, 9:1408–1424.

Lütfi Kerem Şenel, Furkan Şahinuç, Veysel Yücesoy, Hinrich Schütze, Tolga Çukur, and Aykut Koç. 2022. Learning interpretable word embeddings via bidirectional alignment of dimensions with semantic concepts. *Information Processing & Management*, 59(3):102925.

Lütfi Kerem Şenel, Ihsan Utlu, Veysel Yücesoy, Aykut Koc, and Tolga Cukur. 2018. Semantic structure and interpretability of word embeddings. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1769–1779.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3407–3412.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and

Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Nishant Subramani, Nivedita Suresh, and Matthew E Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Kellie Webster, Xuezhi Wang, Ian Tenney, Alex Beutel, Emily Pitler, Ellie Pavlick, Jilin Chen, Ed Chi, and Slav Petrov. 2021. Measuring and reducing gendered correlations in pre-trained models.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

Ke Yang, Charles Yu, Yi Fung, Manling Li, and Heng Ji. 2023. Adept: A debiasing prompt framework. In *Proc. Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI2023)*.

Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535.

Charles Yu, Sullam Jeoung, Anish Kasi, Pengfei Yu, and Heng Ji. 2023. Unlearning bias in language models by partitioning gradients. In *Proc. The 61st Annual Meeting of the Association for Computational Linguistics (ACL2023) Findings*.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions. *arXiv preprint arXiv:2304.14293*.

A Broader Impacts

The intended use of this work is to contribute to advancements in fine-grained and efficient control of language generation in AI, with experiments shown on sentiment modulation, political stance adjustment, and language detoxification. We do not aim to create a tool for manipulating public opinion or promoting specific political ideologies, but instead to provide methods for enhancing the reasoning interpretability, and safety of AI language models. Our techniques offer the potential for fine-tuned sentiment adjustment and toxic content mitigation, thereby contributing to more reliable, unbiased, and respectful language generation systems. We would like to emphasize that on the problem of language model toxicity, we limit our model to modeling detoxification only. This encourages positive and socially beneficial usage of our model as well as general language models.

B Formal Statement of Theorem 1

Hidden Markov Models (HMMs) (Baum et al., 1970) is a widely used framework for analyzing discrete stochastic processes. Because of its generality (modeling arbitrary distributions), intuitiveness, and interpretability (containing a structured state space), it has long been used as a primary choice when modeling language distribution. is a discrete stochastic process with a set of n states $\mathbf{S}$ and a set of m observations or emissions $\mathbf{O}$, with arbitrary indexing of $\mathbf{S}$ and $\mathbf{O}$. The time step $t = 0$ distribution is determined by initial state distribution $s_0 \sim \pi$. For each later time step $t \geq 1$, the state transition probabilities are represented by a matrix $\mathbf{T}$, where $T(s, s') = P(s_{t+1} = s'|s_t = s)$ denotes the probability of transitioning from state s to state s'. At each time step one observation o_t is emitted, with the emission probabilities represented by a matrix $\mathbf{B}$, with $B(s, o) = P(o_t = o|s_t = s)$. A sequence of observations can be denoted as $\mathbf{o} = \{o_1, o_2, \dots, o_T\}$. The probability distribution over sequences $\mathbf{o}$ then follows formula:

$$\begin{aligned} &P(o_1, \cdots, o_T; \pi) \\ &\quad = \pi^\top \left(\prod_{t=0}^{T-1} \mathrm{diag}(\mathbf{p}(o_t)) T \right) \mathbf{p}(o_T), \quad (4) \end{aligned}$$

where $\mathbf{p}(o)$ is a $|\mathcal{S}|$-dim vector indicating $P(o \mid s)$ for all states $s \in \mathcal{S}$.

Language Models In generative language models, the sequence is generated word-by-word by a conditional probability $P(o_t \mid o_1, \cdots, o_{t-1})$. The common technique to model this probability is to first calculate the inner product between a contextual vector $\mathbf{c}(o_1, \cdots, o_{t-1})$ and word embeddings $\mathbf{E} = (\mathbf{e}_o, \cdots) \in \mathbb{R}^{d \times |\mathcal{O}|}$, namely, $\mathbf{l} = \mathbf{c}(o_1, \cdots, o_{t-1})^\top \mathbf{E}$. Here, $\mathbf{l}$ is known as the word *logits*, which then usually passes through a softmax operator to get a distribution over words. For simplicity of analysis, in this work, we assume a linear formulation and let conditional probability $P(o_t|o_1, \cdots, o_{t-1}) = \mathbf{c}(o_1, \cdots, o_{t-1})^\top \mathbf{e}_{o_t}$. By the chain rule, multiplying the conditional probabilities will give us the full probability: $\prod_{t=1}^T P(o_t \mid o_1, \cdots, o_{t-1}) = P(o_1, \cdots, o_T)$. We are then interested in the situation where a language model is good enough to represent an equivalent distribution with HMM.

In this study, we aim to model the influence of *conditions* in text generation. This section describes how we incorporate conditions in HMMs. Conventionally, people assume a d-dimensional state representation ϕ_s for every state s, and d-dimensional ψ_o for each observation o, so that they can compute the probabilities $T(s, s') = \phi_s^\top A \phi'_s$, $B(s, o) = \phi_s^\top \psi_o$ and $\pi(s) = \phi_\pi^\top \phi_s$ for some ϕ_π. We also use matrices Φ, Ψ to denote the stacked representations $\Phi = (\phi_s | s \in \mathcal{S}), \Psi = (\psi_o | o \in \mathcal{O})$. Here we introduce an additional *condition* component in state representations, so that ϕ_s can be partitioned into two sub-vectors: $\phi_s = \begin{pmatrix} \phi_{s,\text{semantic}} \\ \phi_{s,\text{condition}} \end{pmatrix}$. Here $\phi_{s,\text{semantic}} \in \mathbb{R}^{d_s}$ represents the d_s-dim semantic information, and $\phi_{s,\text{condition}} \in \mathbb{R}^{d_c}$ the d_c-dim condition information related to state s. Then we assume that the transition probability $T(s, s')$ comes from both semantic relations and conditional similarities between s' and s: $T(s, s') = \phi_{s,\text{semantic}}^\top A' \phi_{s',\text{semantic}} + \phi_{s,\text{condition}}^\top \phi_{s',\text{condition}}$.

We also make the following assumptions regarding the state representations:

Assumption 1. *State representations ϕ also satisfy the following properties:*

1. Values for each dimension are uniformly normalized to a constant: $\forall i \in [1..d], \sum_{s \in \mathcal{S}} \phi_{s,i}^2 = C$.

2. Dimensions are linearly independent: $\forall i, j \in [1..d]$ and $i \neq j$, $\sum_{h \in \mathcal{H}} \phi_{h,i} \phi_{h,j} = 0$.

3. Dimensions are also conditionally independent: if $i, j \in [1..d], k \in [d_s + 1..d]$ are not all the same, $\sum_{s \in \mathcal{S}} \phi_{s,i} \phi_{s,j} \phi_{s,k} = 0$.

The validity of the assumption is discussed in Appendix H. Then, we present the result below, revealing that shifting between conditions is equivalent to a linear transformation in word embedding space.

Theorem 2. *Assume assumption 1 holds. Suppose there are two initial distributions $\pi = \phi_\pi^\top \Phi, \pi' = \phi_{\pi'}^\top \Phi$, so that ϕ_π and $\phi_{\pi'}$ only differ in their condition-parts: $\phi_{\pi,\text{semantic}} = \phi_{\pi',\text{semantic}}$. Also, suppose the elements in $\phi_{\pi,\text{condition}}$ are non-zero. Then there exists a matrix W so that, by transforming word embeddings from E to WE, the LM which originally simulates the text distribution starting with π will now turn to be equivalent to a distribution initiating from π'.*

C Formal Statement and Proof of Theorem 2

To prove Theorem 2, we start by claiming a construction of matrix W. Then we prove that when assumptions 1 hold, W can change each conditional likelihood function from $p(v_i \mid o_1, \cdot, o_{i-1}, \pi)$ to $p(v_i \mid o_1, \cdot, o_{i-1}, \pi')$ up to a constant factor. Finally, by chaining the conditional likelihoods, we see that W can change the sentence-level probability distribution of the HMM from π-initialization to π'-initialization.

Assuming full column-rank for $\mathbf{E}$ and $\mathbf{p}(o)$, we have the following connection between LM and HMM:

Proposition 1. *There exist projection matrices R_1 and R_2 so that $R_1^\top R_2 = I_n$ and*

$$\mathbf{c}(o_1, \cdots, o_{t-1})^\top = \left(\frac{\pi^\top \prod_{t'=1}^{t-1} \mathit{diag}(\mathbf{p}(o'_t)) T}{\pi^\top \left(\prod_{t'=1}^{t-2} \mathit{diag}(\mathbf{p}(o'_t)) T \right) \mathbf{p}(o_{t-1})} \right) R_1^\top,$$

$$\mathbf{e}_o = R_2 \mathbf{p}(o).$$

We first construct a helper matrix $W' = \begin{pmatrix} I_{d_s} & 0 \\ 0 & \Lambda \end{pmatrix}$ so that Λ is diagonal and $W' \phi_{\text{init}} = \phi'_{\text{init}}$. Such a solution exists as we assume $\phi_{\text{init,condition}}$ contains only non-zero values. Then, we can construct the matrix W as $W = R_1^+ \Phi^\top W' \Phi R_2^{+\top}$, where R_1^+, R_2^+ are pseudo-inverse matrices of R_1, R_2, respectively.

Lemma 3. $T \Phi^\top W' \Phi = \Phi^\top W' \Phi T$.

Proof. First, it is easy to see that, by Assumption 1.1 and Assumption 1.2, the representation matrix Φ is row-orthonormal to constant C_2:

$$\Phi\Phi^T = C_2 I_d$$

.

Then we have the following proof:

$$\begin{aligned}
T\Phi^\top W'\Phi =& \Phi^\top \begin{pmatrix} T_s & 0 \\ 0 & I_{d_c} \end{pmatrix} \Phi\Phi^\top W'\Phi \\
=& C_2\Phi^\top \begin{pmatrix} T_s & 0 \\ 0 & I_{d_c} \end{pmatrix} \begin{pmatrix} I_{d_s} & 0 \\ 0 & \Lambda \end{pmatrix} \Phi \\
=& C_2\Phi^\top \begin{pmatrix} T_s & 0 \\ 0 & \Lambda \end{pmatrix} \Phi \\
=& C_2\Phi^\top \begin{pmatrix} I_{d_s} & 0 \\ 0 & \Lambda \end{pmatrix} \begin{pmatrix} T_s & 0 \\ 0 & I_{d_c} \end{pmatrix} \Phi \\
=& \Phi^\top W'\Phi\Phi^\top \begin{pmatrix} T_s & 0 \\ 0 & I_{d_c} \end{pmatrix} \Phi \\
=& W'T
\end{aligned}$$

□

Lemma 4. *$\forall v \in V$, we have that, $\Phi diag(\mathbf{p}(o))\Phi^\top W'\Phi = W'\Phi diag(\mathbf{p}(o))$.*

Proof. To prove this, we first prove that $\Phi\text{diag}(\mathbf{p}(o))\Phi^\top$ has the form $\begin{pmatrix} A & 0 \\ 0 & \Lambda' \end{pmatrix}$, where Λ' is also diagonal. This is equivalent to saying that, for any two one-hot vectors $\mathbf{e}(i), \mathbf{e}(j)$, if $i \in [d_s + 1..d]$ or $j \in [d_s + 1..d]$, then

$$\begin{aligned}
&\mathbf{e}_i^\top \Phi \text{ diag}(\mathbf{p}(o))\Phi\mathbf{e}_j^\top \\
&= \sum_{h\in\mathcal{H}} \phi_{h,i}\phi_{h,j}p(v \mid h) = f_v(i,j)\mathbf{1}(i = j). \quad (5)
\end{aligned}$$

For any $i \neq j$,

$$\begin{aligned}
&\sum_{h\in\mathcal{H}} \phi_{h,i}\phi_{h,j}p(v \mid h) \\
&= \sum_{h\in\mathcal{H}} \phi_{h,i}\phi_{h,j} \sum_{k\in[1..d]} \phi_{h,k}\theta v, k \\
&= \sum_{k\in[1..d]} \theta v, k \sum_{h\in\mathcal{H}} \phi_{h,i}\phi_{h,j}\phi_{h,k} \\
&= \sum_{k\notin\{i,j\}} \theta_{v,k} \sum_{h\in\mathcal{H}} \phi_{h,i}\phi_{h,j}\phi_{h,k} \\
&\quad + \theta_{v,i} \sum_{h\in\mathcal{H}} \phi_{h,i}^2\phi_{h,j} \\
&\quad + \theta_{v,j} \sum_{h\in\mathcal{H}} \phi_{h,i}\phi_{h,j}^2 \\
\text{(Asm. 1.3) } &= 0 + \theta_{v,i} \sum_{h\in\mathcal{H}} \phi_{h,i}\phi_{h,j} \\
\text{(Asm. 1.2) } &= 0 + 0 \\
&= 0
\end{aligned}$$

We then have the following:

$$\begin{aligned}
&\Phi\text{diag}(\mathbf{p}(o))\Phi^\top W'\Phi \\
=& \begin{pmatrix} A & 0 \\ 0 & \Lambda' \end{pmatrix} W'\Phi \\
=& \begin{pmatrix} A & 0 \\ 0 & \Lambda'\Lambda \end{pmatrix} \Phi \\
=& W' \begin{pmatrix} A & 0 \\ 0 & \Lambda' \end{pmatrix} \Phi \\
=& W'\Phi\Phi^\top \begin{pmatrix} A & 0 \\ 0 & \Lambda' \end{pmatrix} \Phi \\
=& W'\Phi\text{diag}(\mathbf{p}(o))
\end{aligned}$$

□

By combining Lemma 3 and 4, we have the following lemma:

Lemma 5.

$$T diag(\mathbf{p}(o))\Phi^\top W'\Phi = \Phi^\top W'\Phi T diag(\mathbf{p}(o))$$

Proof.

$$\begin{aligned}
&T\text{diag}(\mathbf{p}(o))\Phi^\top W'\Phi \\
=& \Phi^\top \begin{pmatrix} T_s & 0 \\ 0 & I_{d_c} \end{pmatrix} \Phi\text{diag}(\mathbf{p}(o))\Phi^\top W'\Phi \\
=& \Phi^\top \begin{pmatrix} T_s & 0 \\ 0 & I_{d_c} \end{pmatrix} W'\Phi\text{diag}(\mathbf{p}(o)) \\
=& \Phi^\top W'\Phi T\text{diag}(\mathbf{p}(o))
\end{aligned}$$

□

Finally, when we apply Lemma 3 and 5 to the language model formulation, we can see that the conditional likelihood function has been steered to:

$$
\begin{aligned}
&\mathbf{p}_W(v_i \mid o_1, \cdots, o_{i-1}; \pi) \\
=&\mathbf{c}(o_1, \cdots, o_{i-1}; \pi) W E \\
=&\frac{\pi^\top T \text{diag}(o_1) T \cdots T \text{diag}(o_{i-1}) T R_1^\top}{\pi^\top T \text{diag}(o_1) T \cdots T \mathbf{p}(o_{i-1})} W R_2 P_O \\
=&\frac{\pi^\top T \text{diag}(o_1) T \cdots T \text{diag}(o_{i-1}) T \Phi^\top W' \Phi}{\pi^\top T \text{diag}(o_1) T \cdots T \mathbf{p}(o_{i-1})} P_O \\
&\text{(Lemma 3)} \\
=&\frac{\pi^\top T \text{diag}(o_1) T \cdots T \text{diag}(o_{i-1}) \Phi^\top W' \Phi T}{\pi^\top T \text{diag}(o_1) T \cdots T \mathbf{p}(o_{i-1})} P_O \\
&\text{(by Lemma 5)} \\
=&\frac{\pi \Phi^\top W' \Phi^\top T \text{diag}(o_1) T \cdots T \text{diag}(o_{i-1}) T}{\pi^\top T \text{diag}(o_1) T \cdots T \mathbf{p}(o_{i-1})} P_O \\
=&\frac{\phi_{\text{init}} W' \Phi T \text{diag}(o_1) T \cdots T \text{diag}(o_{i-1}) T}{\pi^\top T \text{diag}(o_1) T \cdots T \mathbf{p}(o_{i-1})} P_O \\
=&\frac{\phi'_{\text{init}} \Phi T \text{diag}(o_1) T \cdots T \text{diag}(o_{i-1}) T}{\pi^\top T \text{diag}(o_1) T \cdots T \mathbf{p}(o_{i-1})} P_O \\
&\text{(by definition)} \\
=&\frac{\pi' T \text{diag}(o_1) T \cdots T \text{diag}(o_{i-1}) T}{\pi^\top T \text{diag}(o_1) T \cdots T \mathbf{p}(o_{i-1})} P_O \\
&\propto \mathbf{c}(o_1, \cdots, o_{i-1}; \pi') E \\
=&\mathbf{p}(o_i \mid o_1, \cdots, o_{i-1}; \pi')
\end{aligned}
$$

Therefore, the steered conditional likelihood is equivalent to an HMM initiating from π' (up to a normalization constant over vocabulary $\mathcal{O}$). By chaining the conditional likelihood functions, it is easy to see that the actual output sequence distribution is now:

$$
\begin{aligned}
&p_{W,\text{normalized}}(o_1, \cdots, o_L; \pi) \\
=&\prod_{i=1}^{L} \text{normalize}_{\mathcal{O}}(\mathbf{p}_W(v_i \mid o_1, \cdots, o_{i-1}; \pi)) \\
=&\prod_{i=1}^{L} \mathbf{p}(o_i \mid o_1, \cdots, o_{i-1}; \pi') \\
=&p(o_1, \cdots, o_L; \pi')
\end{aligned}
$$

This concludes our proof to Theorem 2.

D Implementation Details

In this paper, we leverage the HuggingFace package[5] and its model checkpoints. To implement LM-Steer, we simply wrap the `self.forward` function of language model transformer's `lm_head`, and inject the computation formula of LM-Steer. In specific, the token logits are replaced w ith $\mathbf{c}^\top (I + \epsilon W) \mathbf{e}_o$, we change the computation order and first compute $\mathbf{c}' = \mathbf{c} + \epsilon W \mathbf{c}$), then compute $\mathbf{c}'^\top \mathbf{e}_o$. We find that this increases computational efficiency in practice and avoids the problem caused by many Transformers sharing input and output word embedding parameters in storage. Another trick we applied in experiments is that, as there is a systematical distribution shift between pre-training corpus and domain-specific dataset (such as detoxification dataset and reviews), we add one "dummy" steer W_{dummy} to fill this overall distribution gap. Therefore, for positive label training, we use model $P_{\epsilon 0 (W + W_{\text{dummy}})}$, and for negative label training, we use model $P_{\epsilon 0 (-W + W_{\text{dummy}})}$. This is where the 3M parameters come from in Table 4.

For optimization, we use Adam optimizer (Kingma and Ba, 2014) with a 1e-2 learning rate and train for 1k steps. The steer matrix W is initialized with a Gaussian distribution of 0 mean and 1e-3 variance. Across all experiments, we run three initial seeds of 0, 1, and 2 for training. When required to generate 25 sentences on each prompt, we use random seeds 0, 1, 2, ..., 24. Our hardware is one single Tesla V-100 GPU with 16GB CUDA memory.

E Hyperparameter Selection

We select the decoding hyper-parameter based on a balance of task performance and generation quality on the detoxification task's dev set. The scores are listed in the table below.

When gradually increasing the steering value, the detoxification success rate increases while generation fluency decreases. To better balance the two ends, we select $5\epsilon_0$ for downstream evaluation, as it does not compromise perplexity too much while achieving decent task performance.

F Details of Transferring LM-Steer to Other Language Models

To transfer an LM-Steer from one LM M_1 to another LM M_2, we notice that LM-Steer essentially

[5] https://huggingface.co

steering value ϵ	**Toxicity**↓		**Fluency**↓	**Diversity**↑		
	Avg. max. toxicity	Toxicity prob.	Output ppl.	Dist-1	Dist-2	Dist-3
GPT-2 (original)	0.527	0.520	25.45	0.58	0.85	0.85
MuCoLa	0.308	0.088	29.92	0.55	0.82	0.83
LM-Steer (ϵ_0)	0.542	0.560	24.20	0.54	0.85	0.86
LM-Steer ($2\epsilon_0$)	0.473	0.388	24.54	0.54	0.84	0.85
LM-Steer ($3\epsilon_0$)	0.419	0.278	24.83	0.54	0.84	0.85
LM-Steer ($4\epsilon_0$)	0.393	0.232	25.43	0.54	0.83	0.84
LM-Steer ($5\epsilon_0$)	0.370	0.198	26.37	0.54	0.83	0.84
LM-Steer ($6\epsilon_0$)	0.343	0.172	27.53	0.54	0.82	0.83
LM-Steer ($7\epsilon_0$)	0.320	0.138	29.12	0.54	0.81	0.82
LM-Steer ($8\epsilon_0$)	0.306	0.118	31.32	0.54	0.80	0.81

Table 8: Results on language model detoxification task dev set by selecting different steering value ϵ.

adds one term $\mathbf{c}^\top W\mathbf{e}_o$ to the logits, where both $\mathbf{c}$ and $\mathbf{e}_o$ can be viewed as residing in word embedding space. Therefore, W can be considered as a similarity matrix in M_1's word embedding space. To use W in M_2, we propose to map M_2's word embedding space to that of M_1 before using W as usual. The process works in 2 steps.

First, we identify a linear mapping from W_2 to W_1's word embedding space. We start with building a list of anchor words. Specifically, we select the top 4k words shared by both vocabularies. We denote the token embedding matrices as E_1', E_2' respectively. Then, we initialize a mapping H with a Gaussian distribution of 1e-3 initial variance, and we apply Adam optimizer 0.01 learning rate for 5k steps. Secondly, After acquiring the mapping matrix H, we map both the context and embedding vectors to $H\mathbf{c}$ and $H\mathbf{e}_o$, respectively. So the additive term for LM M_2 is now $\mathbf{c}^\top H^\top WH\mathbf{e}_o$, which is equivalent to using a steer matrix $H^\top WH$ for the LM M_2.

This mapping process is not precise, as word embeddings between LMS are not linearly associated. So we observe an increased instability in generation if we use large ϵ. Therefore, we reduce the steering value to 0.1 of its original scale, $0.5\epsilon_0$ for generation. This is the setting for getting results in Figure 5a.

G Details of Investigating Interpretability

In Section 5.1, we interpret the weights learned in LM-Steer and list discovered keywords in Table 6. A detailed description of getting these results is as follows. First, we conduct SVD decomposition of steer matrix W. The resulting D matrix can then be interpreted as a ranked list of significant row vectors. We take the first 9 rows and compute their dot products with word embeddings. As the row vector does not tell us which of the 2 directions indicates an increased probability, we select 20 tokens with top dot product and 20 tokens with bottom dot product as two candidate groups. Each group is concatenated to a text sequence and passed to Perspective API, and the group with a larger toxicity value is considered true "keywords". If, however, Perspective API recognizes the language as not English, which happened to rows No. 2, 4, and 6, then we discard this row as they contain mostly symbols and non-English words. Finally, we filter out suffix tokens, and the remaining keywords are listed in Table 6.

H Validity of Assumptions

To verify the validity of the assumptions, we did an experiment for searching for valid HMMs while satisfying the assumption 1. It is trivial to construct valid Ψ as long as a valid Φ can be found. So specifically, we set $d_s = 20$ and $d_c = 1$ to represent a one-conditional HMM. We let $n = 200$ and randomly initialized Φ with Gaussian distribution with variance 1e-3. Then we construct the following objective function

$$\mathcal{L} = \mathcal{L}_{\text{norm}} + \mathcal{L}_{\text{dist}} + \mathcal{L}_{\text{independence}} + \mathcal{L}_{\text{conditional}},$$

where

$$\begin{aligned}
\mathcal{L}_{\text{norm}} &= \sum_i (\sum_{s\in\mathcal{S}} \phi_{s,i}^2 - \frac{1}{dn} \sum_{s\in\mathcal{S},i'} \phi_{s,i'}^2)^2 \\
\mathcal{L}_{\text{dist}} &= \sum_{s,s'} \max(-T(s,s'),0) \\
&\quad + \sum_s (\sum_{s'} T(s,s') - 1)^2 \\
\mathcal{L}_{\text{independence}} &= \sum_{i\neq j} (\sum_s \phi_{s,i}\phi_{s,j})^2 \\
\mathcal{L}_{\text{conditional}} &= \sum_{i,j,k \text{not one value}, k\in[d_c+1,d]} \\
&\quad + (\sum_s \phi_{s,i}\phi_{s,j}\phi_{s,k})^2
\end{aligned}$$

Generally, this objection characterizes the derivation of Φ from the assumptions. We use Adam optimizer with learning rate 1e-3, and ReduceLROnPlateau [6] with patience 100 and reduce factor 0.5. The optimization process lasts 500k steps, starting from random seeds 0, 1, 2, and 3. On all random seeds, the objective function reduces from greater than 1 to less than 1e-5. This indicates that valid HMM solutions satisfy the assumption.

I Comparison with A (Soft) Word Blacklist

First, we explain that a control vector is equivalent to a SWB. This is because by adding a vector to context $c' = c + \epsilon w$, we are equivalently adding a logit bias to each word: $c'^\top E = c^\top E + \epsilon w^\top E$, where $w^\top E$ is the static logit bias vector for each word. Then we point out that theoretically, LM-Steer is more expressive in representing sequence distributions than them, since LM-Steer's formulation $c^\top(I + \epsilon W)E$ can let different words be preferred in different contexts c. Theoretically, there exists a LM-Steer for steering between any two finite-length distributions (with proof below). SWB cannot achieve this (a counterexample below). Intuitively speaking, a blacklist or whitelist uniformly applied at all positions cannot possibly achieve flexible control over the complex language distribution without hurting the generation quality.

I.1 Formal statement of the universality of LM-Steer

Let D_1 and D_2 be two finite-length finite-vocabulary sequence distributions, there exists a context vector function $c(o_1, o_2, \cdots, o_i)$, a word embedding E and a matrix W so that an LM-Steer with $-W$ and $+W$ represents distribution D_1 and D_2 respectively.

Proof. We prove the existence by construction. Let $I(o_1, o_2, \cdots, o_i) \in \mathbb{N}$ be an arbitrary indexing function for all subsequences. With bounded subsequence length, I values are also bounded by a number d. We let $c(o_1, \cdots, o_i) = \sum_o \text{onehot}(I(o_1, o_2, \cdots, o_i, o)) \in \mathbb{R}^d$. For each token o, the word embeddings is e_o such that, for any subsequence $(o_1, o_2, \cdots, o_i)$, $e_o[I(o_1, o_2, \cdots, o_i, o))] = \frac{(D_1+D_2)(o|o_1,o_2,\cdots,o_i)}{2}$. Then W is as follows: for any $(o_1, o_2, \cdots, o_i)$ and token o, $W[I(o_1, o_2, \cdots, o_i, o), I(o_1, o_2, \cdots, o_i, o)] = \frac{(D_2-D_1)(o|o_1,o_2,\cdots,o_i)}{(D_1+D_2)(o|o_1,o_2,\cdots,o_i)}$. It is diagonal. We omit verification due to space limits. □

I.2 3.2 Construction of a counterexample for SWB

Let D_1 be a one-point distribution on sequence "AB", and D_2 be a uniform distribution on sequences "BA", "AB". For any language model representing D_1, there does not exist an SWB that can convert the language model into D_2. Verification of this counterexample is trivial, and we omit it due to space limits.

J Results of LM-Steer on Pythia Family

Pythia [7] is a family of causal language models developed by EleutherAI. Raining in size from 14M to 2.8B, these models provide an excellent testbed for evaluating the effect of language model sizes. The table below shows the performance of LM-Steer applied to the Pythia language models. We can see a trend of better fluency but decreasing detoxification when the model size increases, indicating a greater controlling difficulty and better base quality of larger language models.

K LoRA Configuration

Thanks for suggesting a more comprehensive evaluation. We use the Huggingface Transformers' default implementation of LoRA on GPT-2-large to align with standard practices in the field and compare with our method. To test the effect of configurations, we iterate the LoRA rank over 8, 16, 32, and 64. Following the practice of (Lee et al., 2023),

[6] https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

[7] https://github.com/EleutherAI/pythia

Model	Toxicity↓		Fluency	Diversity↑		
	Max. toxicity	Toxicity prob.	Output ppl.↓	Dist-1	Dist-2	Dist-3
Pythia-14M	0.208	0.04	85.67	0.50	0.84	0.86
Pythia-70M	0.213	0.06	54.84	0.55	0.86	0.87
Pythia-160M	0.223	0.07	35.24	0.54	0.86	0.87
Pythia-410M	0.255	0.13	36.71	0.58	0.86	0.86
Pythia-1B	0.286	0.17	31.34	0.56	0.85	0.86
Pythia-1.4B	0.289	0.15	31.63	0.58	0.86	0.86
Pythia-2.8B	0.328	0.17	32.90	0.51	0.81	0.85

Table 9: Language model detoxification results of LM-Steer with Pythia

we set the alpha scalar equal to the rank. We report LoRA's performance below.

L Incorporating in LoRA

Thanks to its theoretical foundations, LM-Steer is orthogonal to other methods and is intuitively compostable with other control methods. As an example study, we select LoRA to combine with LM-Steer on detoxification with GPT2-large as the backbone. The result is as follows. Combining LoRA with LM-Steer produces a better detoxification score than LoRA alone (although not as good as LM-Steer alone), at the cost of a degraded quality score.

M Results on GPT-J-6B

We go on and evaluate LM-Steer on GPT-J-6B model. We use the same evaluation setup as in the main body. The results are shown in Table 12. We observe that LM-Steer is able to reduce the toxicity of the generations while maintaining the perplexity and the fluency of the generations. We also observe that the diversity of the generations is not affected by LM-Steer.

N Effect of LM-Steer on Instruction Following

We study the effect of LM-Steer on the LMs' performance under prompts. For prompts, we let ChatGPT write 10 strongly positively biased prompts in Table 13. Then, we let GPT-J generate 25 tokens under instructions (i.e., prompting) and compare its generation with and without LM-Steer.

We see that LM-Steer can still steer generation even under a positive prompt while maintaining generation fluency. Without LM-Steer, the sentiment is 94.00. With a positive LM-Steer, the average sentiment increases to 97.20. With negative LM-Steer steering, the average sentiment decreases significantly to 82.80. Example generation are listed in Table 14 and 15.

O Embedding Tuning

We also consider the possibility of tuning the word embeddings of the backbone model. We use the same training procedure as LM-Steer, but instead of tuning the last layer, we tune the word embeddings. We use the same hyperparameters as LM-Steer. The results are shown in Table 16. We observe that the performance is comparable to Soft-Blacklist, but is still inferior to LM-Steer. We believe that this is because the word embeddings are shared across all the layers, and tuning the word embeddings may cause the model to forget the knowledge learned from the pre-training. We will leave the investigation of this direction for future work. Embedding-tuning is also more expensive than LM-Steer, as it requires tuning the word embeddings for the entire vocabulary, while LM-Steer only requires tuning the last layer.

LoRA rank	Max. Toxicity	Toxicity prob.	ppl.	Dist-1	Dist-2	Dist-3
8	0.362	0.257	23.83	0.532	0.839	0.852
16	0.365	0.210	21.11	0.534	0.845	0.855
32	0.351	0.229	26.13	0.529	0.840	0.853
64	0.354	0.257	23.78	0.531	0.840	0.853

Table 10: The performance of LoRA with different ranks on the detoxification dataset.

Method	**Max Toxicity**	**Toxicity prob**	**ppl**	**Dist-1**	**Dist-2**	**Dist-3**
LoRA	0.365	0.210	21.11	0.53	0.85	0.86
LM-Steer	0.238	0.078	28.26	0.55	0.84	0.84
LoRA + LM-Steer	0.258	0.086	41.78	0.54	0.82	0.83

Table 11: Results on incorporating LM-Steer with LoRA.

Method	**Max Toxicity**	**Toxicity prob**	**ppl**	**Dist-1**	**Dist-2**	**Dist-3**
GPT-J-6B	0.364	0.229	18.70	0.55	0.84	0.85
GPT-J-6B + LM-Steer	0.265	0.124	18.26	0.54	0.84	0.85

Table 12: Results of LM-Steer on GPT-J-6B model on language model detoxification.

“Compose a critique of the novel ’Pride and Prejudice’, emphasizing its flaws while maintaining an overall appreciative tone.“
“Write a review of the restaurant ’Le Gourmet’ focusing on areas for improvement, yet with an underlying tone of admiration for its cuisine.“
“Pen a critique about the summer season, pointing out its drawbacks but in a way that overall celebrates its beauty and warmth.“
“Draft a review for the TV show ’Breaking Bad’, highlighting its weaker aspects but still expressing overall enthusiasm for the series.“
“Create a review of the coffee shop ’Bean There’, discussing its shortcomings while still conveying a sense of enjoyment of its atmosphere.“
“Write an appraisal of the classic album ’Abbey Road’ by The Beatles, noting any perceived faults but with a tone that remains reverent of its musical genius.“
“Offer a critique of the play ’Hamilton’, focusing on its less successful elements while still acknowledging its groundbreaking impact.“
“Compose a review of the city of Paris in winter, pointing out the challenges of the season while still capturing the magic of the city during this time.“
“Draft a review of the novel ’1984’ by George Orwell, discussing its more controversial or challenging aspects but in a context of overall admiration.“
“Write an evaluation of the gaming console PlayStation 5, noting its limitations or flaws while still expressing enthusiasm for its technological advancements.“

Table 13: Instructions proposed by ChatGPT.

“Focus of your critique on its plot and characterization rather than its historical accuracy and literary style.”
“(For example: not enough bread, or service was bad, or the bathrooms are dirty, or ...”
“In the early weeks of the season, most of the country was in the grip of an intense summer.”
“This is a good show for all the time it spends on drug dealing but the characters are”

Table 14: Example generations (one each for the first 4 prompts) without LM-Steer

“ the novel ‘ fails to persuade the reader that the characters’in every sense of the verb ...”
“The rest of the review is up to you. you should probably mention how long you’ve...”
“Tt’s difficult to redeem the negative factors of summers, except for its unpredictability.”
“Dull. It’s a cheap improv comedy that’s almost utterly incoherent in an...”

Table 15: Example generations (one each for the first 4 prompts) with negative LM-Steer

Method	Max Toxicity	Toxicity prob	ppl	Dist-1	Dist-2	Dist-3
Embedding Tuning	0.289	0.0952	20.41	0.53	0.84	0.85

Table 16: Results on embedding tuning.