Dynamic Steering With Episodic Memory For Large Language Models

Van Dai Do[1], Quan Hung Tran[2], Svetha Venkatesh[1], Hung Le[1]
[1]Applied AI Initiative, Deakin University, Australia [2]Meta
{v.do,thai.le}@deakin.edu.au

Abstract

Large Language Models (LLMs) exhibit emergent in-context learning (ICL) capabilities, allowing them to adapt to unseen tasks based on example demonstrations. Traditional ICL embeds examples within the prompt, while activation steering, uses a vector derived from examples to guide the latent states of LLMs toward desired behaviors. However, traditional ICL is difficult to control quantitatively and consumes valuable context space. Existing activation steering methods apply a single sentence-level steering vector uniformly across all tokens, ignoring LLMs' token-wise, auto-regressive nature. This coarse control can lead to inconsistencies and suboptimal adjustments during generation. To address this problem, we introduce Dynamic Steering with Episodic Memory (DSEM), a novel training-free framework that aligns LLMs to given demonstrations by steering at the token level conditioned on the input query. DSEM employs a key-value memory to store associations between generated tokens and steering vectors. During inference, it uses a nearest-neighbor mechanism to dynamically compute steering vectors for each token chunk, enabling more precise and adaptive guidance. Our method surpasses strong baselines across diverse alignment tasks - including safety, style transfer, and role-playing - demonstrating improved alignment as demonstration size scales.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable performance in a wide range of natural language processing tasks, with their effectiveness improving as the number of parameters increases (Brown et al., 2020; Wei et al., 2022; Chowdhery et al., 2023). One of the most intriguing abilities of larger models is In-Context Learning (ICL), where LLMs can perform tasks they have not encountered during training, given only a few demonstrations (Brown et al., 2020). This enables an alternative approach to achieve strong downstream performance by modifying the input prompt in text space, incorporating instructions and in-context examples (Liu et al., 2021).

Despite the intriguing capabilities of ICL, it is highly sensitive to nearly every aspect of the prompt. The selection of demonstrations, the phrasing of instructions, or even subtle variations in example formatting can lead to significant fluctuations in final performance (Liu et al., 2021; Min et al., 2022). This sensitivity raises concerns about the robustness and reliability of ICL in real-world applications. Furthermore, a fundamental limitation of traditional ICL is the restricted context length of most open-source LLMs, which constrains the number of demonstrations that can be included in the prompt. As a result, effectively utilizing ICL requires careful prompt engineering and selection of examples to mitigate these challenges.

One recent approach to address the limitations of traditional ICL is activation steering, which involves modifying the activation layers of LLMs. For instance, Liu et al. (2024) have proposed computing the principal direction of the difference between source and target representations and using this as a single steering vector for all test data. More recently, Wang et al. (2025) introduced a binary masking method to identify the most influential elements in the steering process. A fundamental limitation of these methods is that they are either query-agnostic or lack fine-grained control, hampering final alignment performance. This raises a research question: *How can activation steering be adaptively applied in a query-dependent manner and at the token level to effectively guide LLM generation toward a desired goal?*

To this end, we propose Dynamic Steering with Episodic Memory (DSEM), a novel and efficient steering method designed to steer LLM generations flexibly, enabling quick adaptation to a given input and fine-grained control over the genera-

Findings of the Association for Computational Linguistics: ACL 2025, pages 13731–13749
July 27 - August 1, 2025

tion process. Drawing inspiration from the rapid and instance-based learning mechanisms observed in the hippocampus region of the human brain (Lengyel and Dayan, 2007), our method utilizes an external episodic memory that stores input representations, partially generated outputs, and their stored steering vectors. We use the input-output demonstrations to define the partial outputs and compute the stored steering vectors. During inference, DSEM utilizes the memory to generate steering vectors for each input query. It retrieves stored steering vectors and estimates the chunk-level steering vector using a nearest-neighbor mechanism, ensuring that steering vectors adapt across queries instead of remaining static.

Another significant benefit of DSEM is its ability to compute steering vectors at the token level. Instead of using a fixed steering vector for an entire sentence, DSEM dynamically determines steering vectors for generating the next few tokens. By leveraging token-level information saved in memory (i.e., the partially generated outputs), DSEM retrieves steering vectors that are better aligned with the desired outputs. This approach is motivated by the fact that next-token prediction in transformers (Vaswani et al., 2017) is inherently a token-wise process, suggesting that steering should also be applied at the token rather than the sentence level.

Finally, to preserve the global semantics of the demonstration, DSEM retrieves the most relevant steering vector at inference time and interpolates it with a global steering vector during each token generation step. This interpolation ensures a smooth transition between context-sensitive adjustments and overarching guidance, preventing abrupt shifts in model behavior. The interpolation weight is determined adaptively based on the uncertainty of the memory prediction, allowing DSEM to balance local specificity with global coherence.

In summary, our main contributions are: **(1)** We propose **DSEM**, a fine-grained activation steering method that dynamically interpolates between memory-based and global steering vectors, enabling precise control over generation. **(2)** We propose an uncertainty-driven interpolation mechanism that dynamically balances local specificity with global coherence. **(3)** We empirically demonstrate that DSEM outperforms strong baselines, including fine-tuning and state-of-the-art activation steering methods, across multiple tasks such as detoxification, formality adjustment, and role-playing, as well as across different model architectures and demonstration sizes.

2 Methods

2.1 Problem formulation

In-context Learning. In the standard framework of ICL, we have access to the base language model $\mathcal{L}$, which is trained on large-scale data, and a set of demonstration data $\mathcal{D}_{\text{demos}} = \{(x_i, y_i)\}_{i=1}^{|\mathcal{D}_{\text{demos}}|}$, where each pair (x_i, y_i) represents an input-output example for the task we want to accomplish. Given a test query x_{test}, the input prompt for traditional ICL is constructed by concatenating the demonstration examples with the test query. Specifically, $\textit{Prompt} = concat((x_{\text{demos}}, y_{\text{demos}}), x_{\text{test}})$, where $(x_{\text{demos}}, y_{\text{demos}})$ are the pairs of demonstration examples, and $concat(.,.)$ denotes the concatenation operation. As a result, the model's output is given by $y_{\text{test}} = \mathcal{L}(\textit{Prompt})$. The response y_{test} can be influenced by the demonstration examples provided and their order (Do et al., 2024; Pham et al., 2025). This allows for a degree of control over the output, as different choices or arrangements of demonstrations can lead to varying model behaviors.

Activation Steering. LLMs rely on the Transformer architecture (Vaswani et al., 2017), where self-attention mechanisms relate different token positions in a sequence. In ICL, demonstration examples are preappended to the query, influencing how attention is distributed across tokens. In latent space, Park et al. (2024) found evidence that concepts are represented as linear directions in the activation space of neural networks. Activation steering typically produces a steering vector, which is a modification to the latent space of an LLM that guides its output toward a desired outcome.

2.2 Steering With Episodic Memory

In this section, we present the architecture of our episodic memory $\mathcal{M}$, which extracts the steering vector at the token level during LLM generation. Assume we generate l tokens at a time; the goal is to find a steering vector for those l tokens and steer them toward the desired behavior. For example, in a Shakespearean role-playing task, a source sentence could be *'I need your help now.'* If fully steered toward Shakespearean formality, it would be *"I beseech thee, assist me this very moment!"*, which is unnecessarily elaborate. Steered for clarity, it would remain *'I need help now.'* By balancing both clarity and Shakespearean formality, the final output would be: *'I beseech*

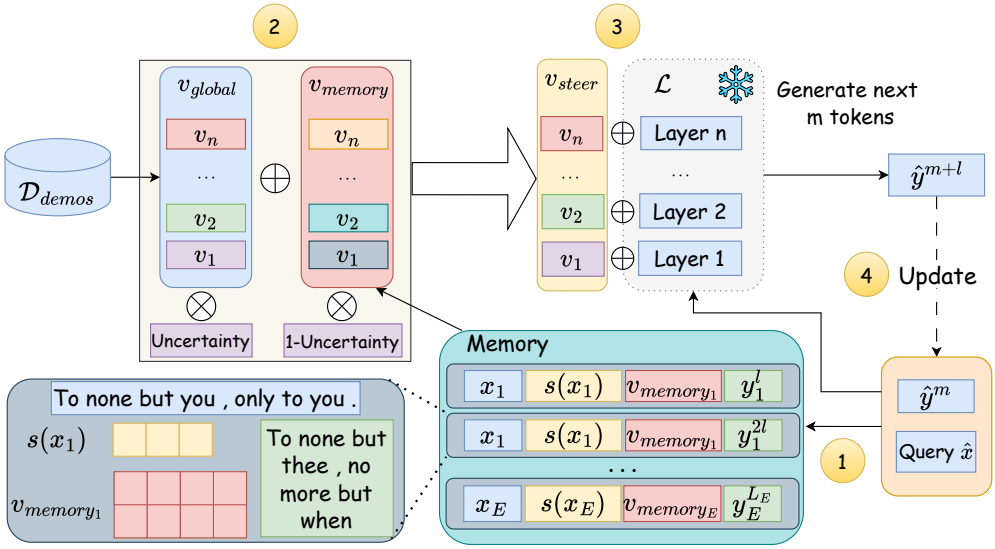

Figure 1: Memory Read process of DSEM and an example element from $\mathcal{M}$ for the Shakespeare dataset (bottom left). Given the input $\hat{x}$ and the current partial generation $\hat{y}^m$, at inference time, Memory is queried with $\hat{x}$ and $\hat{y}^m$ to extract v_{memory} and Uncertainty. v_{memory} undergoes an interpolation phase with v_{global}, derived from $\mathcal{D}_{\text{demos}}$, with interpolation weights determined by the Uncertainty of Memory's prediction. The resulting vector, v_{steer}, is then used to guide the generation of the next l tokens. After that, we do an update on the partially generated output $\hat{y}^m \leftarrow \hat{y}^{m+l}$. This loop is continued until the end of generation.

thee, assist me now.' This shows how having different steering vectors during generation can achieve the optimal balance of style and clarity. In this paper, we utilize an episodic memory to dynamically compute steering vectors in a query-dependent and token-level manner, avoiding the need for expensive fine-tuning (Hu et al., 2021). To write into memory, we store tuples of the demonstration source x, its representation $s(x)$, the corresponding steering vector v_{memory}, and progressively larger token chunks $y^l, y^{2l}, \ldots, y^L$, where L is the target sentence length. During inference (Memory Read phase), the memory $\mathcal{M}$ is queried with the input $\hat{x}$ and partial generation $\hat{y}^m$ to extract v_{memory} and its associated Uncertainty. The extracted vector is interpolated with v_{global} from $\mathcal{D}_{\text{demos}}$ using Uncertainty-based weights, producing v_{steer}. This vector guides the generation of the next l tokens, after which the partial output is updated as $\hat{y}^m \leftarrow \hat{y}^{m+l}$. The process repeats until the generation is completed. An illustration of the process is shown in Figure 1.

State Representation. Accurate and meaningful text representations are essential for both memory storage and efficient retrieval. In DSEM, a memory element contains the embedded representation of the demonstration example x_{demo} along with chunk-level information. For sentence-level information, we compute the mean pooling of the last token representation of the demonstration sentence x_{demo} when passed through the LLM $\mathcal{L}$, which we denote as $s(x_{demo})$. To preserve token-level information, we store the target sentence's tokens progressively. This is achieved by segmenting y_{demo} into $\lceil \frac{L}{l} \rceil$ parts, transforming a sentence of length L into $\lceil \frac{L}{l} \rceil$ progressive states. Each state grants DSEM access to granular token-level insights, enhancing retrieval precision and efficiency.

Steering Vector Extraction. Given a pair of demonstrations $(x_{\text{demo}}, y_{\text{demo}})$, we calculate the steering vector as follows:

$$v_{\text{demo}} = h(y_{\text{demo}}) - h(x_{\text{demo}}) \quad (1)$$

Here, $h(y_{\text{demo}})$ and $h(x_{\text{demo}})$ are the concatenated embeddings across all layers of the LLM $\mathcal{L}$. $v_{\text{demo}} \in \mathbb{R}^{n_{layers}, n_{hidden}}$, where n_{layers} and n_{hidden} denote the number of layers and the latent representation dimension of $\mathcal{L}$, respectively. Intuitively, v_{demo} steers the model's output from x_{demo} toward y_{demo}. Adding v_{demo} to the latent representation of $\mathcal{L}(x_{\text{demo}})$ at each token step guides the model toward generating y_{demo}.

2.2.1 Memory Construction

We populate the memory $\mathcal{M}$ using a set of demonstration examples $\mathcal{D}_{\text{demos}}$. For each example, we pair its text input x_{demo} and embedded representation $s(x_{\text{demo}})$ with the corresponding target out-

put y_{demo}. The output y_{demo} is divided into progressively larger chunks, where the size of each chunk is determined by l. Specifically, the first chunk contains l tokens, the second chunk contains $2l$ tokens, and so on. A smaller l captures finer-grained contextual information but increases memory storage, while a larger l results in coarser chunks, reducing the memory storage. The target sequence is partitioned into multiple chunks for a given l. Each chunked representation is paired with its corresponding steering vector v_{demo}, which is derived from the demonstration. These tuples are then stored in memory $\mathcal{M}$ in the form: $\left(s(x_{\text{demo}}), x_{\text{demo}}, y^j_{\text{demo}}, v_{\text{demo}}\right)$, where y^j_{demo} represents the j-th chunk of the target sequence. These elements are appended sequentially to $\mathcal{M}$, ensuring that all relevant contextual information is preserved.

2.2.2 Memory Read

In this section, we describe the memory reading process used to obtain token-level steering vectors

Sentence-level Retrieval With Reciprocal Rank Fusion. In practice, we often encounter novel sentences, i.e., sentences not present in memory. To address this, we leverage our memory $\mathcal{M}$ as a nearest-neighbor estimator to obtain steering vectors at the token level. We first retrieve $\mathcal{N}_{sent}$, the set containing the top-K_{sent} most relevant sentences. Inspired by Reciprocal Rank Fusion (RRF) (Cormack et al., 2009), we rank these neighbors by combining dense retrieval and BM25 (Robertson and Zaragoza, 2009) to enhance retrieval precision.

For retrieval, we combine dense and lexical similarity to capture both semantic and lexical information. Dense retrieval ranks candidates ($\mathcal{R}_{dense}$) based on their cosine similarity with the test query, capturing semantic alignment. Lexical similarity is introduced by computing BM25 scores between the test query x_{test} and all queries in $\mathcal{M}$, resulting in the ranking R_{BM25}. The details of R_{dense} and R_{BM25} are provided in Appendix A.2.

Once we obtain two rankings, R_{dense} and R_{BM25}, the final rank of each sentence x_i in $\mathcal{M}$, denoted as $R_{RRF}(x_{test})$, is computed using the RRF score:

$$score_{\text{RRF}}(x_i) = \frac{1}{k_{\text{RRF}} + R_{\text{dense}}(x_i)} + \frac{1}{k_{\text{RRF}} + R_{\text{BM25}}(x_i)} \tag{2}$$

where k_{RRF} is a constant integer. Intuitively, a higher $score_{\text{RRF}}$ indicates greater similarity between the memory element x_i and the test query x_{test} at the sentence level. Finally, we use R_{RRF} to get $\mathcal{N}_{sent}$ corresponding to x_{test}.

Token-level Vector Extraction. Beyond retrieving the nearest input neighbors, we also search for the stored target outputs closest to the current generation to refine the steering vector at the token level. This step is performed during generation between the current partial generation and the target chunks we stored in $\mathcal{M}$, which differs from the RRF retrieval used for retrieving the nearest sentences.

Let $\mathcal{N}_{\text{tok}}$ represent the set of chunks derived from the top-K_{sent} retrieved sentences for the test query. Our objective is to retrieve the top-K_{tok} nearest chunks from $\mathcal{N}_{\text{tok}}$ to the current generation. To achieve this, we calculate the BM25 score between the current generation y^m and each chunk y_{tok}, $y_{\text{tok}} \in \mathcal{N}_{\text{tok}}$. The top-$K_{\text{tok}}$ memory elements are then selected based on these scores. This approach provides access to fine-grained token-level information about the current generation with respect to the memory elements, thereby improving the accuracy of the steering vector calculation. For simplicity, we set $K_{\text{tok}} = K_{\text{sent}}$.

Steering Vector Interpolation With Memory Uncertainty. In this section, we illustrate how the token-level steering vector is calculated. First, we show how to extract the memory steering vector, denoted as v_{memory}. When the generation process has not started, it is not possible to use Token-level Retrieval, thus, for that case, we rely on the top-K_{sent} sentences' steering vector. Formally, we have:

$$v_{\text{memory}} = \begin{cases} \sum_{j=1}^{|\mathcal{N}_{\text{tok}}|} v_j \,/\, |\mathcal{N}_{\text{tok}}|, & \text{if } t = 0, \\ \sum_{j=1}^{|\mathcal{N}_{\text{sent}}|} v_j \,/\, |\mathcal{N}_{\text{sent}}|, & \text{otherwise.} \end{cases} \tag{3}$$

Here, t is the number of generated tokens in the response, v_j is the steering vector corresponding to the j^{th} memory element in $\mathcal{N}_{\text{tok}}$.

To balance the tradeoff between local relevance and global coherence during token-level steering, we propose a dynamic interpolation mechanism between memory-based and global steering vectors. The general form of the interpolation is defined as:

$$v_{\text{steer}} = (1 - \delta(x_{\text{test}})) \cdot v_{\text{memory}} + \delta(x_{\text{test}}) \cdot v_{\text{global}}, \tag{4}$$

where v_{global} represents the global steering vector, and $\delta(x_{\text{test}})$ is the uncertainty-based weight that adjusts the contribution of each component.

The memory-based steering vector, v_{memory},

provides localized guidance by capturing token-relevant information from similar past contexts, ensuring improved contextual relevance. However, solely relying on memory can cause the model to overfit to specific memory instances, resulting in outputs that are less consistent with the broader data distribution. To address this, we include the global steering vector, v_{global}, which captures corpus-wide information and aligns with the reference data's statistical properties.

The uncertainty-based interpolation weight $\delta(x_{\text{test}})$ is motivated by two key goals. First, it reduces hyperparameter tuning costs by dynamically adjusting the interpolation weight, eliminating the need for manual tuning of fixed weights and making the approach more adaptive and efficient. Second, it enables dynamic adjustment for each input by varying the interpolation weight based on the uncertainty of the memory retrieval. For inputs with low uncertainty, v_{memory} is prioritized, otherwise, the mechanism shifts focus toward v_{global}.

To compute the global steering vector v_{global}, we adopt the state-of-the-art method from ICV (Liu et al., 2024), defined as:

$$v_{\text{global}} = \arg\max_{h} \frac{1}{|\mathcal{D}|} \sum_{i} \left(h^T(hy_i - hx_i)\right)^2, \quad \text{s.t.} \quad h^T h = 1. \tag{5}$$

Here, y_i and x_i represent the target and source sentences in the demonstration set $\mathcal{D}_{demos}$, respectively. This ensures that v_{global} captures the overall statistical patterns of the corpus, complementing the localized guidance provided by v_{memory}.

To determine the interpolation weight dynamically at the token level, we introduce an uncertainty-based mechanism. Let d be the cosine distances between $s(x_{\text{test}})$ and all stored elements in $\mathcal{M}$. If $s(x_{\text{test}}) \in \mathcal{M}$, the memory provides a direct steering vector with zero uncertainty $\delta(x_{\text{test}}) = 0$. Otherwise, we estimate uncertainty based on the dispersion of distances among the top-K_{sent} retrieved sentences in $\mathcal{M}$: $\delta(x_{\text{test}}) = \frac{\text{mean}(d) - \text{min}(d)}{\text{max}(d) - \text{min}(d)}$. Higher uncertainty δ signals greater variability in memory distances, indicating lower confidence in the retrieved steering vector.

After computing v_{steer} in Equation 4 using the elements outlined above, we apply it to guide the generation of the next l tokens as follows:

$$\hat{v}^i_{\mathcal{L}} = v^i_{\mathcal{L}} + \lambda \cdot v^i_{steer} \tag{6}$$

where i denotes layer i^{th} of $\mathcal{L}$ and λ denotes steering intervention strength. By combining v_{memory} and v_{global} in this framework, we achieve a balance between local adaptation and global coherence at the token level. The uncertainty-based weighting mechanism ensures robust, contextually adaptive steering while minimizing manual effort. Memory Read algorithm is shown in Appendix A.1.

3 Experiments

Our experiments focus on open-source LLMs: **Llama-2-7b-chat**, **Gemma-2-9b-it**, **Falcon-7B**, and **Mistral-7b-Instruct-v0.2** (Touvron et al., 2023; Team, 2024; Almazrouei et al., 2023; Jiang et al., 2023). The same top_p and temperature values are used across all methods within each model.

3.1 Datasets

ParaDetox. We evaluate DSEM on the detoxification task using ParaDetox (Logacheva et al., 2022), a dataset consisting of toxic-neutral sentence pairs. For evaluation, we randomly select 700 examples. The goal is to rewrite toxic sentences into non-toxic ones while preserving their original meaning. **GYAFC.** We evaluate DSEM on the formality transfer task using GYAFC (Rao and Tetreault, 2018), where each sample contains an informal and a formal sentence. The objective is to rewrite informal sentences into their formal counterparts. The dataset is further divided into two subsets: GYAFC-family and GYAFC-music, which contain sentences related to family relationships and music, respectively. **Shakespeare.** We also consider the Shakespeare Modern English dataset (Jhamtani et al., 2017), where each sample consists of a Modern English sentence. The task is to rewrite it in the style of Shakespearean English.

3.2 Baselines

We compare our approach with state-of-the-art steering and finetuning methods across different LLMs and demonstration sizes. **No Context** only uses the test input without additional context. **ICL** adds a set of randomly selected examples from the demonstration set together with the test query to build the prompt. **LoRA** fine-tunes the model with Low-Rank Adaptation (Hu et al., 2021) using the demonstration set as training data. **SaDI** (Wang et al., 2025) averages the differences of all data pairs in the demonstration set. It then finds the most influential elements to create a binary mask to steer the model. **ICV** (Liu et al., 2024) computes a

	Llama-2-7b						Gemma-9b					
	Paradetox		GYAFC_family		GYAFC_music		Paradetox		GYAFC_family		GYAFC_music	
Method	Tox.↓	BERT↑	For.↑	BERT↑	For.↑	BERT↑	Tox.↓	BERT↑	For.↑	BERT↑	For.↑	BERT↑
No Context	39.00	90.75	39.77	92.19	40.56	92.25	44.28	83.82	28.71	82.55	27.51	80.11
ICL-4	41.43	85.09	26.45	55.13	45.33	64.93	22.14	90.24	60.60	92.27	85.36	89.56
E = 10												
ICL-10	25.85	41.03	51.59	66.69	53.62	70.77	14.00	88.98	83.11	90.57	83.42	90.22
ICV	24.86	91.12	50.28	91.34	66.84	91.67	6.71	85.50	51.78	86.61	76.19	85.46
SaDI	28.14	45.50	53.66	92.11	57.14	92.23	78.29	**90.93**	8.07	92.06	5.82	91.38
LoRA	39.71	90.70	50.84	92.04	19.71	90.70	82.71	90.59	7.31	**92.18**	5.64	**91.43**
DSEM	**21.42**	**91.68**	**66.04**	**92.26**	**76.37**	**92.26**	**5.57**	88.29	**96.44**	90.35	**96.47**	90.59
E = 100												
ICV	18.42	90.77	69.36	91.50	55.73	91.60	5.67	84.62	85.37	86.24	78.13	83.97
SaDI	17.86	90.44	52.91	92.10	59.44	92.15	80.43	**91.76**	7.69	**92.60**	6.80	90.13
LoRA	47.14	**93.23**	34.14	**93.77**	32.98	**93.70**	61.29	89.54	5.25	90.27	5.82	**94.42**
DSEM	**15.70**	90.90	**85.00**	92.11	**83.60**	92.22	**4.29**	88.15	**97.75**	90.29	**94.53**	90.41

Table 1: **DSEM vs. Baselines on Detoxification and Formality Transfer.** Results (percentages) across Llama-2-7b and Gemma-9b models. "Tox." is for Toxicity, "For." is Formality. ICL-4 and ICL-10 use 4 and 10 random demonstrations. Lower Toxicity is better, while Higher Formality and BERTScore indicate better performance. Best results are **bolded**, second-best underlined, excluding *No Context* and *ICV-4*.

steering vector based on the principal direction of differences between positive and negative examples from the demonstration set.

3.3 Standard Benchmarking Evaluation

Following prior work (Liu et al., 2024), to measure alignment effectiveness, we use a safety classifier to detect toxicity in ParaDetox generations, trained on datasets like Wikipedia Toxic Comments (Wulczyn et al., 2017), Build It - Break It - Fix It (Votipka et al., 2020), and Bot-Adversarial Dialogue (Xu et al., 2021). A sentence is considered safe if its predicted score exceeds 0.9. The dataset is split into GYAFC-family and GYAFC-music, aligning with subdomains. For formality transfer, we use a RoBERTa-based classifier (Dementieva et al., 2023) with a 0.9 threshold. Additionally, ground-truth relevance is measured using ROUGE-1 (Lin, 2004) and BERTScore (Zhang et al., 2020) for lexical and semantic similarity, respectively. Due to space restriction, the results of detoxification, formality transfer and BERTScore for **Llama-2-7b** and **Gemma-9b** are shown in Table 1, while the full results, including ROUGE-1 metric and other LLMs such as **Falcon-7b** and **Mistral-7b** are shown in Table 5 in Appendix A.4.

Paradetox Results. Our method, DSEM, achieves the lowest toxicity across nearly all models and demonstration sizes, demonstrating its effectiveness in controlled text generation. As shown in Table 1, DSEM consistently outperforms other methods in reducing toxicity while maintaining competitive fluency and relevance. Notably, DSEM achieve the highest scores across all datasets with $E = 10$, which highlights its efficiency over other methods. On Llama-2-7b at $E = 100$, DSEM lowers toxicity to 15.70%, outperforming ICV (18.42%) and significantly surpassing LoRA (47.14%). Although LoRA achieves the highest BERTscores in some settings (e.g., 93.23 for Llama-2-7b at $E = 100$), its high toxicity makes it unsuitable for detoxification tasks. DSEM balances toxicity reduction with coherence and relevance, offering scalable and fine-grained control. For Gemma-9b, DSEM produces meaningful low-toxicity content, whereas ICV, despite similarly low toxicity, generates repetitive and nonsensical tokens (see Appendix A.12).

Formality Transfer Results. DSEM consistently achieves the highest Formality scores across all models and datasets, significantly outperforming baselines. This advantage is most pronounced at $E = 100$, increasing the chance of retrieving better neighbors. With Gemma-9b, performances of DSEM nearly achieve the perfect Formality rates for both datasets. In addition, the performance gaps of DSEM compared to second best baselines are large (e.g., +12.38% on GYAFC_family and +16.40 % on GYAFC_Music while having higher BERTScores with $E = 100$). At $E = 10$, DSEM remains superior. With Llama-2, it achieves 66.04% (GYAFC_family) and 76.37% (GYAFC_music), outperforming second-best baselines, which are SaDI and ICV, by 12.38% and 9.53%, respectively. Notably, these gains come with higher BERTScore, indicating effective memory-based steering.

3.4 LLM Evaluation

To capture subtle aspects of language quality that conventional metrics may overlook, we evaluate using CohereAI's Command-R Plus model, a powerful enterprise LLM with 104 billion parameters

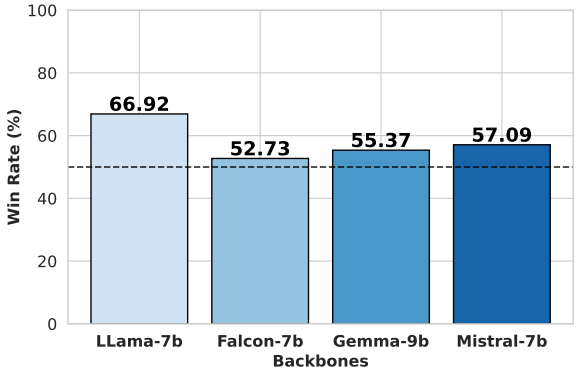

Figure 2: Win rates of DSEM versus ICV across different backbones on the Shakespeare role-playing task using Command-r-plus as evaluator.

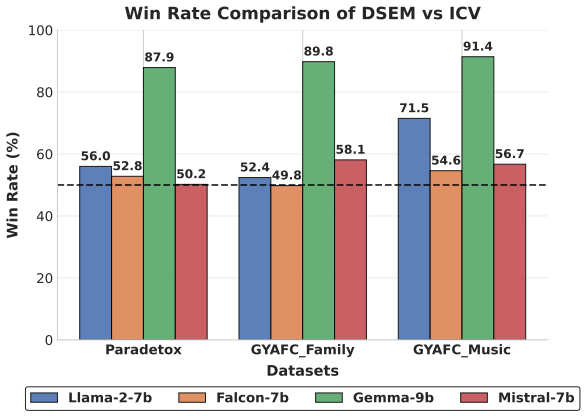

Figure 3: Win rates of DSEM versus ICV across different backbones on other datasets with 300 samples each using Command-r-plus as the evaluator.

(Cohere For AI, 2024). This model acts as a proxy for human evaluation, determining which responses are superior. We randomize the order of the responses in the prompt to avoid possible positional bias (Zheng et al., 2023). Evaluation prompts are reported in Appendix A.16.

Shakespeare Role-Playing Results. We evaluate DSEM on 585 test samples from the Shakespeare role-playing dataset (Jhamtani et al., 2017), using ICV (Liu et al., 2024) as the baseline due to its strong performance in alignment as shown in the previous experiment. We set $E = 100$ and $K_{tok} = K_{sent} = 9$. As shown in Figure 2, DSEM consistently outperforms ICV across all tested backbones. This demonstrates the advantage of our fine-grained steering method in improving response quality while maintaining alignment with the intended role.

Results on Other Datasets. We randomly select 300 samples from each dataset for LLM evaluation

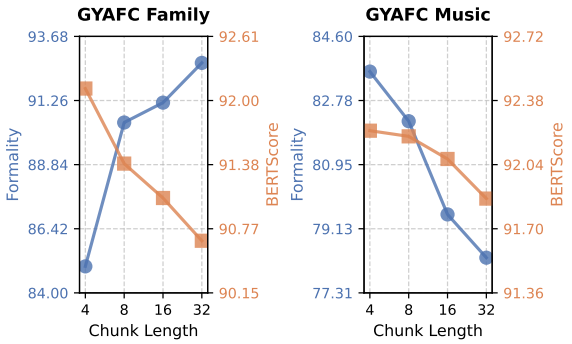

Figure 4: Ablation on using different l with Llama-2-7b and $E = 100$ for GYAFC Family and GYAFC Music.

to further validate the effectiveness of DSEM. We compare the generations head-to-head with ICV (Liu et al., 2024), and the results are presented in Figure 3. DSEM outperforms ICV across all backbone models on the GYAFC_Music dataset, with Gemma-9b achieving the highest win rate of 91.4%. For Paradetox, DSEM consistently performs better across all models. On the GYAFC_Family dataset, except for Falcon-7b, DSEM maintains superiority over ICV, with Mistral-7b achieving the second-best win rate of 58.1%.

4 Ablation Study

4.1 Effect of Different Token Chunk Length l

We study the impact of chunk length l on performance to understand the benefit of token-level steering. Results for Llama-2-7b across GYAFC Family and GYAFC Music datasets are shown in Figure 4. Overall, a lower l tends to boost relevance, as evidenced by higher BERTScores. However, the optimal chunk length varies by dataset. For GYAFC Music, shorter chunks yield substantial gains because they capture fine-grained contextual nuances essential for music-related texts. In contrast, for GYAFC Family, longer chunks better preserve broader context and narrative flow, leading to improved downstream performance. These findings indicate that the optimal l is dataset-dependent, reflecting the varying linguistic and structural characteristics across tasks. Therefore, it is important to steer at a flexible token level and adjust the chunk length to suit different settings.

4.2 Effect of Using Memory Steering

We validate the results when using only v_{memory} as the steering vector, without interpolating with v_{global}. We use Llama-2-7b on the GYAFC Family

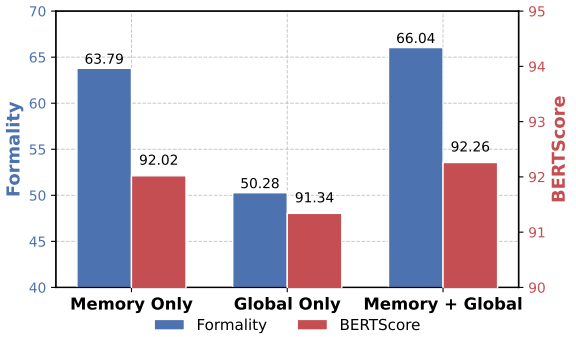

Figure 5: Ablation of only using v_{memory} using Llama-2-7b on GYAFC Family dataset with $E = 10$.

Dataset	Scores	R-1	BERT
Paradetox ↓	16.28 (+1.1)	50.66	90.81
GYAFC_Family ↑	84.99 (-0.01)	45.60	91.47
GYAFC_Music ↑	83.25 (-0.35)	51.25	92.22

Table 2: Results of DSEM using the average of all vectors stored in $\mathcal{M}$ as v_{global}. We use Llama-2-7b with $E = 100$. For Paradetox, Lower score is better (↓), while Higher score is better for the rest (↑).

dataset. For the "Global Only" baseline, v_{global} is derived from ICV (Liu et al., 2024). As shown in Figure 5, using only v_{global} achieves the lowest results among all approaches. Using only v_{memory} yields better results on both Formality and BERTScore. Yet, the best performance is achieved by interpolating between v_{memory} and v_{global}, ensuring smooth transitions between context-sensitive adjustments and overarching guidance.

4.3 Effect of Different v_{global}

In addition to using v_{global} as ICV, we try v_{global} as the average of stored steering vectors in $\mathcal{M}$. Results for Llama-2-7b with $E = 100$ are shown in Table 2. Toxicity increases slightly (1.1%), while Formality decreases marginally. While using v_{global} as ICV performs slightly better, this experiment shows that DSEM is robust to simpler methods of deriving v_{global}.

4.4 Chunk Length Selection

In addition to the results we have already presented in the paper using LLaMA-2-7B, we report additional experiments on the GYAFC_Family and GYAFC_Music datasets using Gemma-9B. The experiments were run with $l \in [2, 4, 8, 16, 32]$.

Observing the results, we find that increasing l generally decreases BERTScore while enhancing the formality of the output, which aligns with our earlier findings using LLaMA-2. However, on the GYAFC_Family dataset, formality continues to increase with larger l, diverging from the trend presented in Figure 4 . Overall, a medium value of l (between 4 and 16) appears to provide a good balance between downstream performance and relevance, whereas values that are too small (e.g., $l = 2$) or too large (e.g., $\ell = 32$) tend to yield suboptimal results.

Dataset	l	Formality	BERT
GYAFC_family	2	96.42	90.25
	4	96.44	90.35
	8	97.18	90.17
	16	97.74	90.21
	32	97.18	90.12
GYAFC_music	2	95.22	90.59
	4	96.47	90.59
	8	96.29	90.62
	16	96.29	90.49
	32	97.17	90.41

Table 3: Results of DSEM using Gemma-9b with different chunk length l.

4.5 Other Ablation Studies

Due to the page limit, we provide more ablation studies in the Appendix. We provide additional results on larger LLMs in Appendix A.5's Table 6, results on TruthfulQA dataset in Appendix A.6's Table 9, ablation with various number of neighbors in Appendix A.7's Figure 8, ablation on using DSEM with more demonstration examples in Appendix A.8 's Table 7, ablation on the intervention factor in Appendix A.9's Table 9, inference speed and memory in Appendix A.10's Table 10 and Table 12. These results demonstrate DSEM's consistent outperformance across various settings.

5 Related Work

Traditional ICL Improvements. Several methods have been proposed to improve ICL, including instruction tuning, demonstration selection, and ordering (Deng et al., 2022; Zhang et al., 2022; Do et al., 2024; Pham et al., 2025). Instruction tuning fine-tunes models on diverse tasks to enhance generalization, while demonstration selection identifies the most relevant examples to optimize performance. Ordering strategies explore how the arrangement of examples impacts output. While these approaches enhance ICL by editing instructions or selecting and arranging examples, they remain constrained by the fixed context window, limiting scalability.

Activation Steering refers to techniques that modify the latent space of models to produce desired outputs. Burns et al. (2022); Park et al. (2024) demonstrate that latent knowledge encoded in the activation space is linearly separable. Liu et al. (2024) proposes finding the principal direction of the steering vectors in a demonstration set and applying it to the entire test set. Wang et al. (2025) introduces a binary mask for the latent space to enhance the values of the most important elements. However, while Liu et al. (2024) identifies a query-agnostic steering vector, Wang et al. (2025) relies heavily on brute-force search to determine the optimal number of important elements and the strength of the intervention, a limitation acknowledged in their work. In contrast, DSEM introduces a dynamic, query-dependent approach to identify steering vectors at the token level, allowing the steering direction to adapt smoothly during generation.

6 Conclusion

We introduce Dynamic Steering with Episodic Memory (DSEM), a training-free framework that aligns LLMs with given demonstrations by steering at the token level, conditioned on the input query. DSEM stores representation-vector associations and uses a nearest-neighbor mechanism to extract the steering vector during generation. It leverages prediction uncertainty to determine the interpolation weight with a global steering vector. Evaluations on safety, style transfer, and role-playing tasks demonstrate its effectiveness, with detailed model analyses highlighting its advantages.

Limitations

We explore the ability to steer LLM generations using Dynamic Steering with Episodic Memory (DSEM). Our work introduces query-dependent, token-level steering during generation, a promising direction that warrants further research attention. However, several limitations must be acknowledged. First, while DSEM demonstrates promising results on medium-scale datasets, its scalability to larger datasets remains to be evaluated. Second, verifying memory-based generation poses challenges in black-box models where internal representations are inaccessible. Future research should aim to develop techniques for better understanding these behaviors and extend our findings to broader contexts.

Acknowledgments

This work was supported by compute credits from a Cohere Labs Research Grant, these grants are designed to support academic partners conducting research with the goal of releasing scientific artifacts and data for good projects.

References

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *Preprint*, arXiv:2311.16867.

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.

Collin Burns, Haotian Ye, Dan Klein, and Jacob Steinhardt. 2022. Discovering latent knowledge in language models without supervision. *ArXiV*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Cohere For AI. 2024. c4ai-command-r-plus-08-2024.

Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, page 758–759, New York, NY, USA. Association for Computing Machinery.

Daryna Dementieva, Nikolay Babakov, and Alexander Panchenko. 2023. Detecting text formality: A study of text classification approaches. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages 274–284, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Dai Do, Quan Tran, Svetha Venkatesh, and Hung Le. 2024. Large language models prompting with episodic memory. In *Proceedings of the 27th European Conference on Artificial Intelligence 2024*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Harsh Jhamtani, Varun Gangal, Eduard Hovy, and Eric Nyberg. 2017. Shakespearizing modern language using copy-enriched sequence-to-sequence models. *Preprint*, arXiv:1707.01161.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Máté Lengyel and Peter Dayan. 2007. Hippocampal contributions to control: the third way. *Advances in neural information processing systems*, 20.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What makes good in-context examples for gpt-3? *Preprint*, arXiv:2101.06804.

Sheng Liu, Haotian Ye, Lei Xing, and James Y. Zou. 2024. In-context vectors: Making in context learning more effective and controllable through latent space steering. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 32287–32307. PMLR.

Varvara Logacheva, Daryna Dementieva, Sergey Ustyantsev, Daniil Moskovskiy, David Dale, Irina Krotova, Nikita Semenov, and Alexander Panchenko. 2022. ParaDetox: Detoxification with parallel data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6804–6818, Dublin, Ireland. Association for Computational Linguistics.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *Preprint*, arXiv:2202.12837.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2024. The linear representation hypothesis and the geometry of large language models. *Preprint*, arXiv:2311.03658.

Kha Pham, Hung Le, Man Ngo, and Truyen Tran. 2025. Rapid selection and ordering of in-context demonstrations via prompt embedding clustering. In *The Thirteenth International Conference on Learning Representations*.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality 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 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Gemma Team. 2024. Gemma.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

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

Daniel Votipka, Kelsey R. Fulton, James Parker, Matthew Hou, Michelle L. Mazurek, and Michael W. Hicks. 2020. Understanding security mistakes developers make: Qualitative analysis from build it, break it, fix it. In *USENIX Security Symposium*.

Weixuan Wang, JINGYUAN YANG, and Wei Peng. 2025. Semantics-adaptive activation intervention for LLMs via dynamic steering vectors. In *The Thirteenth International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama,

Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. *Preprint*, arXiv:1610.08914.

Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2021. Bot-adversarial dialogue for safe conversational agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2950–2968, Online. Association for Computational Linguistics.

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. 2022. Tempera: Test-time prompting via reinforcement learning. *arXiv preprint arXiv:2211.11890*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

A Appendix

A.1 Pseudocode for Memory Read of DSEM

We provide the pseudocode for Memory Read process in Algorithm 1

Algorithm 1 Token-level Steering With Memory Read

Require: Memory $\mathcal{M}$, Language Model $\mathcal{L}$, Token step l, Demo set $\mathcal{D}_{demos}$, Test sample x_{test}, Max tokens L, End-of-sequence token <eos>

Initialize: $y^t \leftarrow y^0$, $t \leftarrow 0$, termination $\leftarrow$ False
Compute v_{global} via Eq. (5)
Obtain $s(x_{\text{test}})$
while not termination **do**
  Compute R_{dense}, R_{BM25}
  Compute R_{RRF} via Eq. (2), retrieve $\mathcal{N}_{\text{sent}}$
  if $t = 0$ **then**
    Retrieve $\mathcal{N}_{\text{tok}}$ from top-K_{sent} (based on R_{RRF})
    $v_{\text{memory}} \leftarrow \frac{1}{|\mathcal{N}_{\text{tok}}|} \sum_{j=1}^{|\mathcal{N}_{\text{tok}}|} v_j$
  else
    $v_{\text{memory}} \leftarrow \frac{1}{|\mathcal{N}_{\text{sent}}|} \sum_{j=1}^{|\mathcal{N}_{\text{sent}}|} v_j$
  end if
  Interpolate $v_{\text{steer}} = (1 - \delta(x_{\text{test}})) \cdot v_{\text{memory}} + \delta(x_{\text{test}}) \cdot v_{\text{global}}$
  Add v_{steer} at all layers of $\mathcal{L}$: $\hat{v}^i_{\mathcal{L}} = v^i_{\mathcal{L}} + \lambda \cdot v^i_{steer}$
  Generate next l tokens: $y^{t:t+l} \leftarrow \mathcal{L}_{\text{steered}}(y^t, x_{\text{test}})$
  Update sequence: $y^t \leftarrow \text{concat}(y^t, y^{t:t+l})$
  Update number of generated tokens: $t \leftarrow t + l$
  termination $\leftarrow (t \geq L)$ or (<eos> $\in y^{t:t+l}$)
end while

A.2 Dense Retrieval and BM25 Scoring Details

We show detailed formulas for Dense Retrieval and BM25 retrieval here. For dense retrieval, we retrieve $\mathcal{R}_{dense}$ for a test query x_{test} as the dense retrieval ranking based on their cosine distances with the test query:

$$d_{cs}(s(x_{\text{test}}), s(x_i)) = 1 - \frac{s(x_{\text{test}}) \cdot s(x_i)}{\|s(x_{\text{test}})\| \|s(x_i)\|} \quad (7)$$

where $s(x_i)$ is a latent representation of demonstration source x_i saved in $\mathcal{M}$.

To complement dense retrieval, we rank memory elements using BM25, which captures lexical

	Llama-2-7b			Gemma-9b			Falcon-7b			Mistral-7b		
Method	Tox. ↓	R-1 ↑	BERT↑	Tox.↓	R-1↑	BERT↑	Tox. ↓	R-1↑	BERT ↑	Tox. ↓	R-1↑	BERT↑
No Context	39.00	52.11	90.75	44.28	34.20	83.82	89.14	67.75	92.04	31.28	38.78	89.66
ICL - 4	41.43	38.48	85.09	22.14	52.14	90.24	84.00	70.67	92.91	22.43	49.77	90.17
E = 10												
ICL - 10	25.85	29.35	41.03	14.00	52.61	88.98	85.26	34.64	85.67	24.71	**59.94**	89.82
ICV	24.86	55.09	91.12	6.71	36.15	85.50	**32.14**	63.17	91.76	15.57	38.04	89.43
SaDI	28.14	45.50	90.48	78.29	**63.66**	**90.93**	84.00	70.72	92.91	22.57	34.52	89.26
LoRA	39.71	46.47	90.70	82.71	62.38	90.59	85.14	**70.98**	92.95	26.38	42.72	**90.10**
DSEM	**21.42**	**56.64**	**91.68**	**5.57**	27.37	88.29	32.57	65.05	**92.98**	**12.29**	35.68	89.98
E = 100												
ICV	18.42	50.47	90.77	5.67	25.60	84.62	26.43	57.19	91.52	11.71	33.32	88.61
SaDI	17.86	45.08	90.44	80.43	**67.02**	**91.76**	84.14	**70.99**	**92.98**	23.00	34.54	89.23
LoRA	47.14	**70.74**	**93.23**	61.29	61.94	89.54	58.29	68.61	92.38	33.28	**70.64**	**93.28**
DSEM	**15.70**	**50.97**	90.90	**4.29**	27.70	88.15	**21.42**	56.98	92.75	**9.71**	33.54	88.99

Table 4: **Model Comparison on Detoxification Task.** Results (percentages) for various methods. "Tox." indicates Toxicity; ICL-4 and ICL-10 use 4 and 10 random demonstrations, respectively. ICL with $E = 100$ is omitted due to context length limits. For ICV, LoRA, and DSEM, results are shown for different E. Lower Toxicity scores are better, while higher ROUGE-1 (R-1) and BERT scores indicate better relevance. Best results are **bolded**, second-best underlined. Notation excludes *No Context* and *ICV-4*.

overlap. Given a test query x_{test}, the BM25 score for a memory element x_i is computed as:

$$\text{BM25}(x_i, x_{\text{test}}) = \sum_{t_j \in x_{\text{test}}} \frac{\text{IDF}(t_j) \cdot f_{t_j,x_i} \cdot (k_1 + 1)}{f_{t_j,x_i} + k_1 \cdot (1 - b + b \cdot \frac{|x_i|}{\text{avgdl}})} \tag{8}$$

where f_{t_j,x_i} is the term frequency of t_j in x_i, and $|x_i|$ is its word count. The average length of elements in $\mathcal{M}$ is denoted as avgdl. Hyperparameters k_1 and b control term frequency saturation and length normalization, respectively.

The inverse document frequency (IDF) of a term t_j is given by:

$$\text{IDF}(t_j) = \log\left(\frac{N - n_{t_j} + 0.5}{n_{t_j} + 0.5} + 1\right) \tag{9}$$

where N is the total number of memory elements, and n_{t_j} is the count of elements containing t_j. We rank memory elements based on their BM25 scores and fuse rankings with dense retrieval using RRF.

A.3 Addtional results: Toxicity.

We present the complete results of four LLMs: **Llama-2-7b**, **Gemma-9b**, **Falcon-7b**, and **Mistral-7b** on the Paradetox dataset. The results are summarized in Table 4. As shown in the table, DSEM achieves the best Toxicity scores in all but one setting, demonstrating the effectiveness of the dynamic steering it provides. For instance, with **Mistral-7b**, DSEM achieves a 12.29% Toxicity score while ranking second in terms of BERTScore. Similarly, for **Llama-2-7b**, DSEM achieves the best scores across all metrics—Toxicity, ROUGE-1, and BERTScore—with $E = 10$. Regarding relevance, while DSEM lags behind SaDI and LoRA in certain settings, it is important to note that their Toxicity scores are significantly higher. This trade-off indicates that their relevance scores become less meaningful, as such an imbalance is suboptimal.

A.4 Addtional results: Formality.

We provide Formality results of **Falcon-7b** and **Mistral-7b** in Table 5. With Mistral, DSEM achieves near-perfect scores (96.62% for GYAFC_family, 96.30% for GYAFC_music) with $E = 100$. A similar trend is observed when $E = 10$. **Mistral-7b** outperforms ICV (the second best baselines) in both GYAFC_Family and GYAFC_Music by 8.82 % and 7.58 % while having higher relevant scores. This further indicates the superior in steering control of DSEM.

On GYAFC_music, DSEM more than doubles the third-best Formality score (39.68% vs. 15.87%). Even losing to ICL-10 in $E = 10$ setting, DSEM improves significantly when $E = 100$. We suspect that with more demonstration examples, DSEM is more likely to find suitable steering vector via closer neighbors, thus illustrating better performances.

	Llama-2-7b						Gemma-9b					
	GYAFC_family			GYAFC_music			GYAFC_family			GYAFC_music		
Method	For.	R-1	BERT	For.	R-1	BERT	For.	R-1	BERT	For.	R-1	BERT
No Context	39.77	49.60	92.19	40.56	52.34	92.25	28.71	30.42	82.55	27.51	29.34	80.11
ICL-4	26.45	34.72	55.13	45.33	41.51	64.93	60.60	50.66	92.27	85.36	42.71	89.56
E = 10												
ICL-10	51.59	43.63	66.69	53.62	46.54	70.77	83.11	51.74	90.57	83.42	48.73	90.22
ICV	50.28	46.40	91.34	66.84	48.89	91.67	51.78	33.47	86.61	76.19	22.92	85.46
SaDI	53.66	47.18	92.11	57.14	49.54	92.23	8.07	**61.61**	92.06	5.82	60.72	91.38
LoRA	50.84	46.75	92.04	19.71	46.50	90.70	7.31	61.08	**92.18**	5.64	**61.69**	**91.43**
DSEM	**66.04**	**47.90**	**92.26**	**76.37**	**50.05**	**92.26**	**96.44**	30.07	90.35	**96.47**	37.25	90.59
E = 100												
ICV	69.36	48.54	91.50	55.73	54.71	91.6	85.37	26.29	86.24	78.13	19.91	83.97
SaDI	52.91	46.71	92.13	59.44	49.66	92.15	7.69	61.57	**92.60**	6.80	61.06	90.13
LoRA	34.14	**61.45**	**93.77**	32.98	**64.24**	**93.70**	5.25	**62.34**	90.27	5.82	**69.70**	**94.42**
DSEM	**85.00**	51.69	92.11	**83.60**	51.71	92.22	**97.75**	30.15	90.29	**94.53**	35.45	90.41
	Falcon-7b						**Mistral-7b**					
	GYAFC_family			GYAFC_music			GYAFC_family			GYAFC_music		
Method	For.	R-1	BERT	For.	R-1	BERT	For.	R-1	BERT	For.	R-1	BERT
No Context	4.70	61.58	92.73	6.52	60.40	91.72	65.67	35.42	90.80	63.32	40.45	90.89
ICL-4	40.71	27.09	84.81	45.15	35.74	86.16	51.59	42.72	80.00	71.43	45.65	81.53
E = 10												
ICL-10	**41.78**	54.47	91.78	**41.62**	55.18	90.91	74.86	**54.22**	90.85	75.66	**52.74**	90.15
ICV	17.07	66.27	93.82	15.87	66.08	93.51	83.49	30.89	89.93	88.89	39.44	90.57
SaDI	4.88	64.05	93.26	5.82	62.24	92.10	67.73	36.66	90.91	68.08	41.80	91.02
LoRA	3.75	64.34	93.38	5.29	63.21	92.34	50.84	44.11	**91.75**	26.29	42.79	90.10
DSEM	34.90	**67.42**	**93.90**	39.68	**69.02**	**94.23**	**92.31**	32.76	90.66	**96.47**	40.74	**91.32**
E = 100												
ICV	55.16	66.09	93.99	32.09	66.25	93.40	93.99	33.29	90.22	92.59	39.21	90.52
SaDI	7.69	61.57	92.60	6.80	61.06	90.13	65.48	36.37	90.87	68.25	42.17	91.06
LoRA	27.58	**68.74**	**94.54**	25.57	**69.70**	**94.42**	21.39	**62.23**	**92.96**	23.81	**65.26**	**93.64**
DSEM	**64.43**	66.87	94.02	**48.85**	69.32	94.16	**96.62**	35.36	90.88	**96.30**	40.97	91.28

Table 5: Comparison of different methods on Paradetox and two formality transfer datasets using all 4 LLMs: **Llama-2-7B**, **Gemma-9b**, **Mistral-7b** and **Falcon-7B** with $E = 10$ and $E = 100$. "Tox." denotes Toxicity, "For." denotes Formality, "R-1" represents ROUGE-1, and "BERT" refers to BERTScore. Lower values are better for Toxicity, while higher values are better for all other metrics. In this table, for DSEM, $K_{tok} = K_{sent} = 5$ for all runs. Best downstream performance results (Toxicity and Formality) are **bolded**, with second-best results underlined.

A.5 Additional results: Performance comparison on larger LLMs.

We provide results on the Paradetox, GYAFC_Family, and GYAFC_Music datasets using larger LLMs: meta-llama/Llama-2-13b-chat-hf and Falcon-11B (Touvron et al., 2023; Almazrouei et al., 2023). Results can be seen in 6. Our approach is compared against two state-of-the-art baselines: ICV (Liu et al., 2024) and SaDI (Wang et al., 2025). The detailed results are presented in Table 6. It is evident that DSEM consistently outperforms other methods across the majority of tasks, particularly excelling on GYAFC_Family and GYAFC_Music. For instance, with Llama-2-13B, DSEM achieves an impressive 95.68% formality rate at $E = 100$, while still maintaining high BERTScore and ROUGE-1 scores, demonstrating its effectiveness in preserving both formality and content quality. This highlights DSEM's robustness in structured text transformation. For Paradetox, DSEM continues to deliver strong performance. Notably, it achieves the best downstream results across 3 datasets with Falcon-11B at $E = 10$ and remains highly competitive at $E = 100$, securing the best on GYAFC datasets and second-best performance on Paradetox. These results suggest that DSEM is particularly effective in leveraging larger models and dataset-specific characteristics, providing strong improvements over existing baselines. These results suggest the ability to scale up to larger LLMs of DSEM, proposing a rapid and

	Llama-2-13B								
	Paradetox			GYAFC_Family			GYAFC_Music		
Method	Tox.	R-1	BERT	For.	R-1	BERT	For.	R-1	BERT
E = 10									
ICV	20.71	**56.18**	91.43	83.49	34.12	90.04	65.08	50.11	92.06
SaDI	**17.28**	41.44	90.02	60.41	**42.72**	**91.49**	63.49	45.74	91.65
DSEM	19.86	54.40	**91.61**	**87.42**	37.89	90.81	**73.37**	**51.20**	**92.59**
E = 100									
ICV	15.43	46.71	90.13	93.62	37.83	90.54	**85.71**	41.09	90.46
SaDI	15.88	41.91	90.07	60.40	42.71	**91.49**	63.69	45.64	91.66
DSEM	**14.89**	**47.21**	**90.09**	**95.68**	**42.81**	91.29	81.83	**49.35**	**92.23**
	Falcon-11B								
	Paradetox			GYAFC_Family			GYAFC_Music		
Method	Tox.	R-1	BERT	For.	R-1	BERT	For.	R-1	BERT
E = 10									
ICV	25.43	54.48	91.51	63.41	**60.71**	**93.28**	87.47	49.47	92.18
SaDI	31.71	49.36	91.02	64.92	48.64	92.63	62.43	**52.37**	**92.64**
DSEM	**21.29**	**55.18**	**91.63**	**81.61**	53.52	92.32	**88.00**	49.31	91.86
E = 100									
ICV	**15.42**	46.43	90.49	90.68	36.71	90.78	90.59	34.15	89.98
SaDI	33.14	**49.77**	**91.01**	66.23	**48.45**	**92.61**	63.84	**53.53**	**92.76**
DSEM	16.42	47.55	90.60	**94.93**	45.84	91.45	**92.42**	46.41	91.46

Table 6: Comparison of different methods on Paradetox and two formality transfer datasets using **Llama-2-13B** and **Falcon-11B** with $E = 10$ and $E = 100$. "Tox." denotes Toxicity, "For." denotes Formality, "R-1" represents ROUGE-1, and "BERT" refers to BERTScore. Lower values are better for Toxicity, while higher values are better for all other metrics. In this table, for DSEM, $K_{tok} = K_{sent} = 5$ for all runs. Best downstream performance results (Toxicity and Formality) are **bolded**, with second-best results underlined.

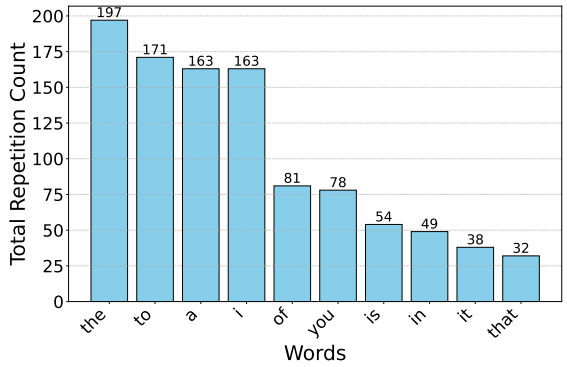

Figure 6: DSEM top 10 most repeated words across generation responses

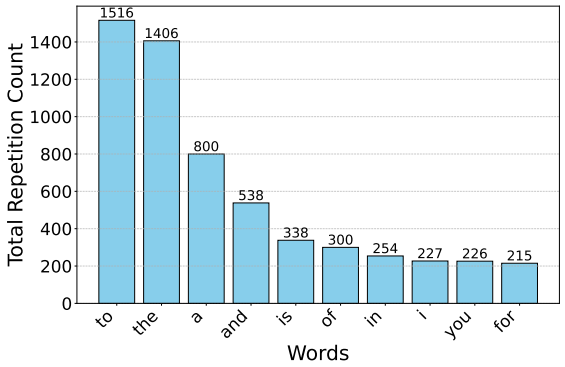

Figure 7: ICV top 10 most repeated words across generation responses

model-free approach to enhance larger LMs performances.

A.6 Additional results: TruthfulQA

We evaluate DSEM on *TruthfulQA*, a dataset designed to assess whether LLMs generate truthful answers. We sample 700 examples from its test set for evaluation. A head-to-head comparison is conducted between DSEM and *ICV* and reported in Table 9, which we consider the strongest baseline based on previous results in this paper. To determine which response is better, we employ *Cohere-r-plus* as the judge. To demonstrate the generalizability of our improvement, we evaluate across four different model backbones. For all runs, we use $E = 100$ samples and $K_{tok} = K_{sent} = 9$. As shown in the results, DSEM outperforms the current state-of-the-art method, ICV, across all settings. Notably, with Gemma-9B, DSEM achieves an impressive 87.51% win rate, significantly surpassing

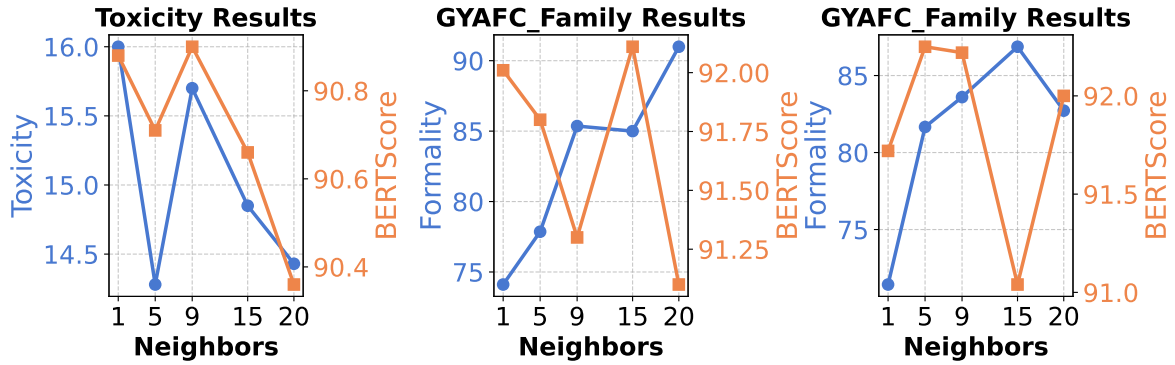

Figure 8: Downstream results (Toxicity, Formality) and BERT scores of DSEM on Paradetox and GYAFC datasets with various number of neighbors (1, 5, 9, 15, 20), using Llama-2-7b with $E = 100$. We note that for simplicity, we use $K_{tok} = K_{sent}$ in all our experiments.

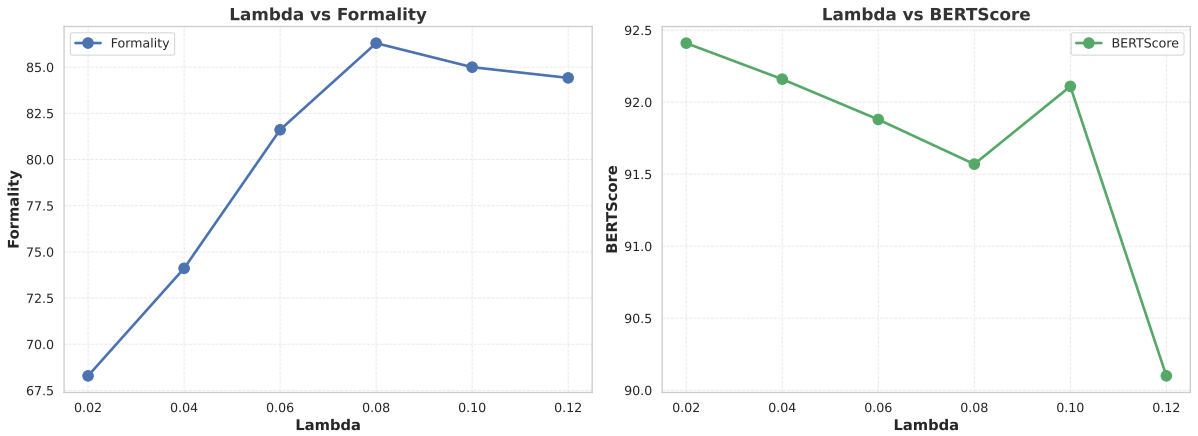

Figure 9: Results of DSEM with different values of λ. We use Llama-2-7b with $E = 100$ on GYAFC Family dataset.

Dataset	Scores	R-1	BERT
Paradetox (↓)	4.00	27.54	88.05
GYAFC Family (↑)	98.87	26.85	89.78
GYAFC Music (↑)	96.82	31.42	89.67

Table 7: Comparison (in percentages) of DSEM using Gemma-9b($E = 1000$) examples. For Paradetox, a lower score is better (↓), while for GYAFC datasets, a higher score is better (↑).

Method	Parameters	Value
LoRA	Number of epochs	3
	Cutoff Length	1024
	Preprocessing workers	16
	Per device train batch size	1
	bf16	True
	Learning rate	$1e^{-4}$
	Learning rate scheduler type	cosine
	Gradient accumulation step	8
	Warm up ratio	0.1
ICV	Intervention strength (λ)	0.1
SaDI	K	6
	Intervention strength (δ)	5

Table 8: Baselines training details

ICV and further demonstrating the effectiveness of our approach.

A.7 Ablation: Different number of neighbors

We present results in Figure 8, showing the impact of varying the number of neighbors $(1, 5, 9, 15, 20)$ on DSEM performance using Llama-2-7b with $E = 100$ on ParaDetox and GYAFC. For simplicity, we set $K_{sent} = K_{tok}$ in all experiments. ParaDetox achieves optimal performance at $K = 9$, balancing toxicity reduction and semantic preservation, while GYAFC benefits from larger K (15 or 20), despite minor semantic drift. These results underline the task-specific nature of neighbor selection.

Model	E	Vs-baseline	Winrate
Llama-2-7b	100	ICV	54.97
Gemma-9b	100	ICV	87.51
Mistral-7b	100	ICV	56.01
Falcon-7b	100	ICV	54.74

Table 9: Win rates of DSEM versus ICV baseline on TruthfulQA dataset with $E = 100$

A.8 Ablation: More examples

We evaluate the performance of DSEM with $E = 1000$ across three datasets using Gemma-9b. Results can be seen in Table 7. The Toxicity score and the Formality score on GYAFC Music improve compared to settings with lower E, while the relevance scores are only marginally lower. This demonstrates that DSEM is susceptible to noise when more memory elements are introduced, highlighting its scalability challenges.

A.9 Ablation: Different values of intervention weight λ

We study the effect of different values of λ using Llama-2-7b with $E = 100$ on the Formality Family dataset. The results are presented in Figure 9. As λ increases, formality steadily improves, peaking at $\lambda = 0.08$ before slightly declining. BERTScore remains relatively stable across all values, with a small dip observed beyond $\lambda = 0.1$. These findings suggest that $\lambda = 0.1$ achieves the best trade-off, optimizing text quality while maintaining both formality and semantic similarity.

A.10 Ablation: Inference Speed and Memory Usage

We include analysis of inference speed in Table 10 and memory usage of our episodic memory in Table 12. Since these values do not vary significantly across different datasets, we report results only for the *Paradetox* dataset.

Inference Speed. Table 10 compares toxicity rates and running times using *Gemma-9B* across all baselines and settings (*with* and *without* episodic memory) on the *Paradetox* sampled test set (700 samples). It can be seen that *DSEM* achieves a significantly lower toxicity rate compared to other baselines, while also being faster than the primary baseline, *ICV*. This highlights *DSEM*'s effectiveness in producing more aligned outputs while maintaining reasonable speed.

GPU Memory Usage. Table 12 shows the GPU memory consumption (in MB) of our episodic memory. As the usage does not vary significantly across datasets, we report numbers only for the *Paradetox* dataset. As expected, our memory module stores only a limited number of examples (as few as 10), with each stored value being a steering vector. Consequently, our framework does not raise concerns regarding memory overflow.

A.11 Discussion on Quality of Memory Entries

In this section, we discuss how the quality of memory entries can effect the performance of our method. We follow standard practice in in-context learning when using curated data, a setup also adopted by prior state-of-the-art methods (e.g., ICV, SaDI). We do not cherry-pick the demonstrations; instead, we randomly select the in-context examples. The availability of demonstrations is required for both traditional in-context learning and activation steering methods.

A.12 Result Analysis: Paradetox

We provide further analysis in the extreme case of Gemma-9b. As shown in Table 1, with Gemma-9b and $E = 100$, the toxicity rates of ICV and our method DSEM are only approximately 5%, which is very low compared to other baselines. However, upon closer inspection, we notice that the answers produced by ICV contain repeated tokens, which fragment its BERTScore with respect to the reference sentence. We provide both a qualitative example and a quantitative plot of Word Repetition per Generation (Repetition count) (Figure 6 and Figure 7), which highlights how many words appear more than once in each generation. We can see that the word 'to' is repeated by ICV more than 1500 times, and 'the' is used more than 1400 times. In contrast, DSEM illustrates better behavior, with its most common repeated word, 'the', appearing less than 200 times. We also provide an example that shows the repetition of generations of ICV in Figure 10.

A.13 DSEM hyperparameters details.

We provide specific parameters used in DSEM in Table 13. In addition, we also provide the number of neighbors used in all of our experiments. We note that $K_{tok} = K_{sent}$ for simplicity.

E	Baseline	Toxic rate	Run time (hours)
0	Nocontext	44.28	0.20
4	ICL-4	22.14	0.35
	ICL-10	14.00	0.57
	ICV	6.71	0.63
	SaDI	78.29	0.40
	LoRA	82.71	0.28
10	DSEM	5.57	0.60
	ICV	5.67	0.67
	SaDI	80.43	0.45
	LoRA	61.29	0.42
100	DSEM	4.29	0.67

Table 10: Toxic rate and run time of different models under various E values.

Model	Paradetox	Family	Music	Shakespeare
E = 10				
Llama-7b	9	5	5	9
Falcon-7b	5	15	15	9
Gemma-9b	9	9	9	5
Mistral-7b	9	9	9	9
Llama-13b	5	9	9	9
Falcon-11b	9	5	5	9
E = 100				
Llama-7b	9	15	15	9
Falcon-7b	5	15	15	9
Gemma-9b	9	9	9	5
Mistral-7b	9	9	9	9
Llama-13b	5	9	9	9
Falcon-11b	9	5	5	9

Table 11: DSEM neighbors selected in experiments. We note that $K_{tok} = K_{sent}$.

Model	E	Memory Used (MB)
Llama-2-7b	10	131.90
	100	223.86
Falcon-7b	10	113.07
	100	235.07
Gemma-9b	10	133.32
	100	238.53
Mistral-7b	10	131.94
	100	224.02
Falcon-11b	10	140.70
	100	311.42
Llama-13b	10	137.52
	100	280.69

Table 12: Memory used of different models with $E \in [10, 100]$

A.14 Other Baselines details.

To conduct LoRA fine-tuning, we utilize Llama-Factory (Zheng et al., 2024). We also provide important parameters for other baselines in Table 8. We note that for (Wang et al., 2025), we select $K = 6$ as they claim it to be in range of best working parameters.

Parameter	Notation	Value
Chunk size	l	4
Intervention strength	λ	0.1
RRF constant	k_{RRF}	60

Table 13: DSEM parameters

A.15 Hardware requirements.

All of our experiments were conducted on a single NVIDIA H100 GPU. We used pytorch=2.5.1, CUDA=12.1, cuDNN=9.1.0.70 and transformers=4.47.1.

A.16 Evaluation Prompts

We provide the prompts used for LLM evaluations in Figure 11, Figure 12 and Figure 13.

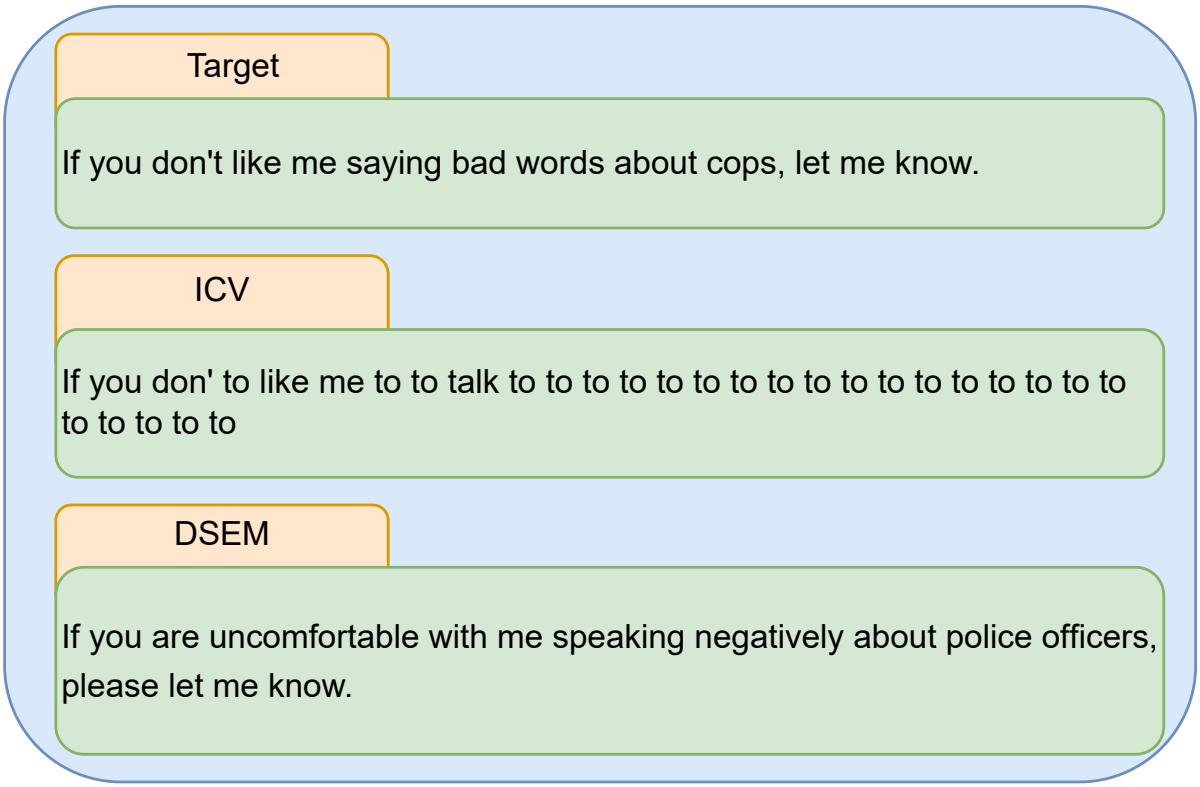

Figure 10: Example of DSEM and ICV generations on Paradetox dataset. From this example, it can be seen that DSEM greatly reduces toxicity, while being relevant to the target sentence. For ICV, the output is often meaningless and repetitive, which, in a way, makes the content non-toxic.

Paradetox

Please act as an impartial and objective judge and evaluate the quality of formality. You should rank the models based on how formal the sentence is, as well as content relevance to reference sentence. The rankings are then output using Python dictionaries and lists. Your evaluation should consider these factors below:
1. Which answer is more formal. The more formal the answer, the better.
2. Which one better preserves the semantic meaning of the reference sentence. The more semantic meaning is preserved, the better.
This is the reference sentence:
{}
This is the formality transfer sentence from model-A:
{}
This is the formality transfer sentence from model-B:
{}
Now, based on the above two criteria, please rank the models. Avoid any positional biases and ensure that the order in which the responses are presented does not influence your decision. Do not favor certain model names. Be as objective as possible. Then, use a list containing the model's name, its rank, and the reason for its ranking to return the results, i.e., please ensure to use the following format to return the results:
[{{"model": <model-name>, "reason": <rank-reason>, "rank": <model-rank>}}, {{"model": <model-name>, "reason": <rank-reason>, "rank": <model-rank>}}]
Your answer must STRICTLY follow the format above, and must be a valid Python list of dictionaries to ensure I can directly parse it using Python. Always use double quotes for the keys and values. Do not include any extraneous content! Please provide a ranking that is as accurate as possible and aligns with the intuition of most people.

Figure 11: Cohere Evaluation Prompt For Paradetox Dataset

GYAFC

Please act as an impartial and objective judge and evaluate the quality of formality. You should rank the models based on how formal the sentence is, as well as content relevance to reference sentence. The rankings are then output using Python dictionaries and lists. Your evaluation should consider these factors below:
1. Which answer is more formal. The more formal the answer, the better.
2. Which one better preserves the semantic meaning of the reference sentence. The more semantic meaning is preserved, the better.
This is the reference sentence:
{}
This is the formality transfer sentence from model-A:
{}
This is the formality transfer sentence from model-B:
{}
Now, based on the above two criteria, please rank the models. Avoid any positional biases and ensure that the order in which the responses are presented does not influence your decision. Do not favor certain model names. Be as objective as possible. Then, use a list containing the model's name, its rank, and the reason for its ranking to return the results, i.e., please ensure to use the following format to return the results: [{{"model": <model-name>, "reason": <rank-reason>, "rank": <model-rank>}}, {{"model": <model-name>, "reason": <rank-reason>, "rank": <model-rank>}}]
Your answer must STRICTLY follow the format above, and must be a valid Python list of dictionaries to ensure I can directly parse it using Python. Always use double quotes for the keys and values. Do not include any extraneous content! Please provide a ranking that is as accurate as possible and aligns with the intuition of most people.

Figure 12: Cohere Evaluation Prompt For GYAFC Datasets

Shakespeare

Please act as an impartial and objective judge and evaluate the quality of the role-playing performance. You should rank the models based on the role characteristics as well as content relevance. The rankings are then output using Python dictionaries and lists. The models below are to play the role of William Shakespeare. Your evaluation should consider these factors below:
1. Which one better preserves the semantic meaning of the reference sentence. The more semantic meaning is preserved, the better.
2. Which one is better at not generating redundant information. The less redundant information is generated, the better.
3. Which one has a more pronounced role-speaking style, and speaks more in line with the reference sentence in terms of the style.
This is the reference sentence:
{}
This is the role-playing sentence from model-A:
{}
This is the role-playing sentence from model-B:
{}
Now, based on the above two criteria, please rank the models. Avoid any positional biases and ensure that the order in which the responses are presented does not influence your decision. Do not favor certain model names. Be as objective as possible. Then, use a list containing the model's name, its rank, and the reason for its ranking to return the results, i.e., please ensure to use the following format to return the results: [{{"model": <model-name>, "reason": <rank-reason>, "rank": <model-rank>}}, {{"model": <model-name>, "reason": <rank-reason>, "rank": <model-rank>}}]
Your answer must STRICTLY follow the format above, and must be a valid Python list of dictionaries to ensure I can directly parse it using Python. Always use double quotes for the keys and values. Do not include any extraneous content! Please provide a ranking that is as accurate as possible and aligns with the intuition of most people.

Figure 13: Cohere Evaluation Prompt For Shakespeare Dataset