Similarity-Distance-Magnitude Language Models

Allen Schmaltz
Reexpress AI
allen@re.express

Abstract

We introduce SIMILARITY-DISTANCE-MAGNITUDE (SDM) language models (LMs), which are sequence prediction models fine-tuned to maximize the proportion of generations in the well-calibrated, high-probability region partitioned by a final-layer SDM activation layer used for binary classification of instruction-following. We demonstrate that existing pre-trained decoder-only Transformer LMs can be readily converted into SDM LMs via supervised fine-tuning, using the final-layer SDM activation layer during training to estimate a change-of-base for a supervised next-token loss over a contrastive input encoding scheme, with additional hard negative examples generated online during training. This results in reduced abstentions (i.e., improved statistical efficiency) compared to strong supervised baselines.

1 Introduction

SIMILARITY-DISTANCE-MAGNITUDE (SDM) activations are a more robust and interpretable formulation of the standard softmax activation function (Schmaltz, 2025). When used as the final-layer activation over the frozen parameters of pre-trained language models (LMs), they provide relatively robust estimators of the predictive uncertainty for selective classification. With this newfound behavior, the focus of modeling shifts to maximizing the proportion of generations in the HIGH-RELIABILITY (HR) region, a partitioning of the output distribution estimated to only contain points with class- and prediction-conditional accuracies at least α, a value near 1 (here, $\alpha = 0.95$).[1]

SDM activations are predicated on the distilled representations of a network; an alternative training approach is needed for learning the weights of the underlying network. Toward this end, we introduce a supervised fine-tuning method that uses a final-layer SDM activation layer for document-level binary classification of instruction-following to estimate a change-of-base for a supervised next-token loss over a contrastive input encoding scheme. We show this achieves the goal of updating the parameters to increase the proportion of generations in the HR region, with improvements over strong supervised baselines that also incorporate our proposed encoding scheme and online generation of hard negatives.

2 Setting

We assume each training example consists of a prompt and a "positive" generation, which we seek for the model to learn to generate, paired with a static (offline) corresponding "negative" generation, which we seek to discourage the model from generating. Unlike the typical setting of post-hoc preference fine-tuning (Schulman et al., 2017; Ouyang et al., 2022; Rafailov et al., 2023, inter alia), we seek to also jointly train the model for binary classification of instruction-following to distinguish positive and negative generations. We seek more substantive changes to the model behavior than surface-level stylistic changes; as such, our points of comparison will be to token-level supervised fine-tuning methods.

Specifically, each example in the training dataset, $\mathcal{D}_{\rm tr} = \{(\boldsymbol{t}_n, \boldsymbol{s}_n^-, \boldsymbol{s}_n^+)\}_{n=1}^N$, consists of an input prompt, $\boldsymbol{t}$, which is a sequence of tokens, paired with positive, $\boldsymbol{s}^+$, and negative, $\boldsymbol{s}^-$, examples that are sequences that do and do not, respectively, follow the instructions of the given prompt. Each $\boldsymbol{s}^+$ and $\boldsymbol{s}^-$ sequence is assumed to end with a special control token representing a positive or negative, respectively, binary classification label (see § 3.1). The LM, π, generates a completion given a prompt, $\hat{\boldsymbol{s}} \sim \pi(\boldsymbol{t})$. To determine the binary classification suffix label of generations created online during training, we assume access to a function,

[1] We assume the notation of Schmaltz (2025) throughout.

$r : (\boldsymbol{s}^{+}, \hat{\boldsymbol{s}}) \mapsto \{0, 1\}$, which provides a binary estimate of whether the generation matches the labeled positive sequence. In the present work, to avoid introducing an additional source of variation in our controlled experiments, we assume r is deterministic (similar to "verifiable rewards" in reinforcement learning), in our case via exact string matches.[2] We assume an analogous labeled calibration dataset, $\mathcal{D}_{\text{ca}}$, drawn from the same distribution as $\mathcal{D}_{\text{tr}}$.

3 Methods

SDM LMs incorporate three unique aspects: (1) A final-layer SDM activation layer for use at test-time for binary classification of instruction-following and at training-time as part of a next-token loss; (2) a contrastive input encoding scheme; and (3) online generation of hard negatives during training.

3.1 Encoding: CONTRASTIVE-MASKING

Each sequence $\boldsymbol{s}$ ends with special control tokens: In the present work, `<verified>Yes</verified>` for sequences that correctly follow the instructions in the prompt, and `<verified>No</verified>`, otherwise. We refer to our encoding scheme as CONTRASTIVE-MASKING (Table 1), because it includes a binary mask for use when calculating the next-token loss during training. In addition to typical masking of the prompt sequence and padding tokens (if applicable), we mask the loss for all tokens in negative sequences up to the final `<verified>No</verified>` control tokens.

3.2 SDM LM Fine-tuning

Training set SDM estimator. During fine-tuning, at the beginning of each epoch, holding the parameters of the underlying network fixed, we train a final-layer SDM activation for binary classification. The SDM layer takes as input $\boldsymbol{t}$ and $\boldsymbol{s}$ up to the rightmost `<verified>` tag.[3] The input to the 1-D CNN is then the concatenation of the final-layer hidden state of the final `<verified>` sequence position with the mean over all final-layer hidden states, as in Schmaltz (2025). The training and calibration split of this SDM layer are constructed each epoch by randomly splitting $\mathcal{D}_{\text{tr}}$.

SDM Next-token Loss. During the epoch, holding the parameters of the SDM layer fixed, we then estimate the predictive uncertainty of each offline example or online generation[4] (see § 3.3) with the document-level SDM activation layer to determine the base for the next-token loss:

$$\begin{aligned}
\mathcal{L} &= -\frac{1}{\sum_{n}^{N} |\boldsymbol{s}_n|} \sum_{n}^{N} \sum_{i}^{|\boldsymbol{s}_n|} \log_{\beta_n} \left(\frac{\beta_n^{z_{i_n}}}{\sum_{v=1}^{|\mathcal{V}|} \beta_n^{z_{v_n}}} \right), \\
\beta_n &= 2 + \text{SDM}(\boldsymbol{z}')_y, \\
y &= r(\boldsymbol{s}_n^{+}, \boldsymbol{s}_n) \qquad (1)
\end{aligned}$$

where i_n is the index of the correct next token in document $\boldsymbol{s}_n$, which is either a generation or a static example from the dataset; z is the output from the final-layer linear layer over the vocabulary, $\mathcal{V}$; and $\text{SDM}(\boldsymbol{z}')_{\hat{y}} \in [0, 1]$ is the estimate of the predictive uncertainty over the document-level binary classification from the SDM activation layer, but for calculating the next-token loss, $\text{SDM}(\boldsymbol{z}')$ is indexed by the output from the function r. (The masking described in § 3.1 is implicit in our notation; in practice, the loss is only calculated over the non-masked tokens of a given $\boldsymbol{s}_n$.) Eq. 1 is the standard cross-entropy next-token loss (as used in our baselines) when $\beta_n = e$, which is constant for all N documents. The loss has the desired semantics of penalizing sequences that are difficult for the SDM layer to classify at the document level, while reducing the cost of token-level errors for sequences with true positive document-level classifications in high probability regions, reflecting the transition from reducible to irreducible error.

Calibration set SDM estimator. Prior to each evaluation over the held-out $\mathcal{D}_{\text{ca}}$, holding the parameters of the underlying network fixed, we train a separate final-layer SDM activation for binary classification. The training and calibration splits of this SDM layer are constructed by randomly splitting $\mathcal{D}_{\text{ca}}$. The lowest SDM next-token loss over $\mathcal{D}_{\text{ca}}$ determines the final model checkpoint.

3.3 Online Generation of Hard Negatives

For the next-token loss and training the SDM activation layers, we use the same proportion of positive and negative sequences. For each prompt, $\boldsymbol{t}_n$, we sample with probability γ^{+} whether the document will be treated as a positive instance instead of a negative instance. For positive instances, the

[2] This is distinct from the probabilistic estimation from the SDM activations of the model undergoing fine-tuning. In typical applications, r is envisioned to be deterministic and/or a combination of SDM activation layers over exogenous models.

[3] The remaining `Yes</verified>` or `No</verified>` tokens are intended to be force-decoded based on the SDM estimator. This convention is for end-users, as well as for test-time search operations that continue conditionally generating.

[4] For documents that were in the training set of the SDM layer, the nearest match is ignored when calculating q and d.

next-token loss is calculated over the static positive example, $\boldsymbol{s}_n^+$, in the dataset. For negative instances, starting after the first epoch, with probability γ^{gen}, we generate a completion, $\hat{\boldsymbol{s}}$. If $r(\boldsymbol{s}^+, \hat{\boldsymbol{s}}) = 0$, with probability $\gamma^{\text{diversity}}$, we use $\hat{\boldsymbol{s}}$ as $\boldsymbol{s}_n^-$; otherwise, we use the static negative example in the dataset.[5] This enables training the model to classify examples known to be wrong, even if rarely generated; to learn to classify hard negatives; and to encourage variation in completions for a given prompt.

3.4 Test-time Generation and Classification

At test-time, greedy generation proceeds as typical, equivalent to using $\beta_n = e$, since the modification of the base does not alter $\arg\max \boldsymbol{z}$ of the final linear layer over the vocabulary. The SDM layer over $\mathcal{D}_{\text{ca}}$ is that used at test-time over unseen data.

4 Experiments

Task. We seek a representative task for which the base LM performs very poorly; can be objectively and quickly evaluated with minimal irreducible error; and requires more abstract skills to solve, such as search and constraint satisfaction, rather than purely static knowledge acquisition. A variation of the classic word ordering task (Elman, 1990; Brown et al., 1990; Brew, 1992), in which the LM is tasked with reordering the final three words of a sentence, fulfills these desiderata (Table 1).

Data. 12,000 examples are sampled from the 7.83 million line subset of processed English Wikipedia in the SentenceTransformers repository[6] (Reimers and Gurevych, 2019) restricted to sentences of length 5 to 60 words. (The mean length is 19.1 words.) $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{\text{ca}}$ each consist of 5,000 examples, and the corresponding in-distribution test set (WIKIPEDIA) is the remaining 2,000 examples.

We also consider a covariate shifted test set (WIKIPEDIACVS) of 2,000 sentences all of length 60, and an in-distribution test set consisting of the 720 classic IEEE "Harvard" sentences[7] (HARVARD) (IEEE, 1969). For all datasets and splits, offline negatives are constructed by randomly permuting the suffix to not match the reference sentence. For 10% of negative examples, we additionally drop (with equal probability) the opening, closing, or both `sentence` XML tags.

Models. We use the `Phi-3.5-mini-instruct` model (PHI3.5), a 3.8 billion-parameter decoder-only Transformer-based language model (Abdin et al., 2024), as our base model. This representative LM is at a scale that enables full parameter fine-tuning of multiple baselines, given available resources, for our controlled study. The parameters of the SDM activation layers are the same as in Schmaltz (2025).

Comparisons. The baseline is the model prior to fine-tuning, PHI3.5. We then compare to fine-tuning with the standard cross-entropy loss using the CONTRASTIVE-MASKING encoding with online hard negatives, PHI3.5+FT(CM+HN), and fine-tuning only using the static examples in the dataset, PHI3.5+FT(CM). For the model fine-tuned with the SDM layers using Eq. 1 and online hard negatives, we use the label PHI3.5+SDM+FT(CM+HN); for the analogous setting with static examples, we use the label PHI3.5+SDM+FT(CM). For fine-tuning, $\gamma^+ = 0.5$, and for generations at training, $\gamma^{\text{gen}} = 0.5$, and $\gamma^{\text{diversity}} = 0.5$. After training is complete, reflecting real-world applications, even for models not fine-tuned with SDM layers and/or online negatives, we train a final SDM layer for use at test-time over $\mathcal{D}_{\text{ca}}$ with $\gamma^+ = 0.5$ and generated hard negatives. However, unlike in training, $\gamma^{\text{gen}} = 1.0$ and $\gamma^{\text{diversity}} = 1.0$, due to the reduced computational costs of a single pass over $\mathcal{D}_{\text{ca}}$ and the relative rarity of hard negatives after fine-tuning. For test-time evaluation, the model is given a prompt and generates a completion, $\hat{\boldsymbol{s}}$, via greedy decoding with $\beta_n = e$, with the ground-truth taken as $y = r(\boldsymbol{s}^+, \hat{\boldsymbol{s}})$. Finally, to get a preliminary sense of variance, we re-run fine-tuning for the full approach a second time, PHI3.5+SDM+FT(CM+HN)$_{\text{2nd}}$.

5 Results

The SDM next-token loss significantly increases the proportion of generations in the HR region for the in-distribution data, as shown in the main results (Table 2). There are no clear differences over the covariate shifted dataset (WIKIPEDIACVS). This is evidence that Eq. 1 has the desired ef-

[5] This basic setup readily extends to the setting of multiple generated completions, or multiple static negative examples. The proposed loss semantics are also compatible with the case of multiple generated positive sequences for which $r(\boldsymbol{s}^+, \hat{\boldsymbol{s}}) = 1$. We omit these cases for presentational simplicity, since they are not examined in the experiments.

[6] https://huggingface.co/datasets/sentence-transformers/wikipedia-en-sentences

[7] https://www.cs.columbia.edu/~hgs/audio/harvard.html

SEQUENCE TYPE	SEQUENCE TOKENS
Prompt (t)	`Complete the sentence 'Neat plans' by reordering all of the following without adding new punctuation nor words: 'fail luck. without'. Only reply with the sentence in the XML <sentence> </sentence> followed by <verified>Yes</verified> if your answer correctly addressed the instructions, and <verified>No</verified> if it did not.`
Negative completion (s^-)	`<sentence>Neat plans without fail luck.\n<verified>No</verified>`
Positive completion (s^+)	`<sentence>Neat plans fail without luck.</sentence>\n<verified>Yes</verified>`

Table 1: Illustrative example prompt and completions for the word ordering task using the CONTRASTIVE-MASKING encoding. The next-token loss is only calculated over the underlined tokens.

Dataset	Model	ACC.	$\frac{n}{\lvert\mathcal{D}_{\text{te}}\rvert}$
HARVARD	PHI3.5	-	0.
HARVARD	PHI3.5+FT(CM)	0.99	0.47
HARVARD	PHI3.5+FT(CM+HN)	0.98	0.36
HARVARD	PHI3.5+SDM+FT(CM)	0.98	0.74
HARVARD	PHI3.5+SDM+FT(CM+HN)	0.97	0.90
HARVARD	PHI3.5+SDM+FT(CM+HN)$_{\text{2nd}}$	0.99	0.90
WIKIPEDIA	PHI3.5	-	0.
WIKIPEDIA	PHI3.5+FT(CM)	0.98	0.56
WIKIPEDIA	PHI3.5+FT(CM+HN)	0.98	0.55
WIKIPEDIA	PHI3.5+SDM+FT(CM)	0.97	0.64
WIKIPEDIA	PHI3.5+SDM+FT(CM+HN)	0.98	0.72
WIKIPEDIA	PHI3.5+SDM+FT(CM+HN)$_{\text{2nd}}$	0.97	0.73
WIKIPEDIACVS	PHI3.5	-	0.
WIKIPEDIACVS	PHI3.5+FT(CM)	1.	<0.01
WIKIPEDIACVS	PHI3.5+FT(CM+HN)	0.88	0.01
WIKIPEDIACVS	PHI3.5+SDM+FT(CM)	1.	<0.01
WIKIPEDIACVS	PHI3.5+SDM+FT(CM+HN)	0.97	0.01
WIKIPEDIACVS	PHI3.5+SDM+FT(CM+HN)$_{\text{2nd}}$	0.94	0.02

Table 2: Selective classification results for instruction-following (content and formatting), among generations in the HR region. A higher proportion of generations in the HR region, $\frac{n}{|\mathcal{D}_{\text{te}}|}$, is preferable, provided the accuracy is $\geq \alpha = 0.95$. For reference comparison, a final-layer SDM layer is learned over all baselines. With PHI3.5, $q'_{\min} = \infty$ (i.e., all generations are rejected).

		SENTENCE	EXACT MATCH
Dataset	Model	ACC.	ACC.
HARVARD	PHI3.5	0.41	0.33
HARVARD	PHI3.5+FT(CM)	0.95	0.94
HARVARD	PHI3.5+FT(CM+HN)	0.95	0.94
HARVARD	PHI3.5+SDM+FT(CM)	0.93	0.93
HARVARD	PHI3.5+SDM+FT(CM+HN)	0.94	0.93
HARVARD	PHI3.5+SDM+FT(CM+HN)$_{\text{2nd}}$	0.96	0.96
WIKIPEDIA	PHI3.5	0.26	0.25
WIKIPEDIA	PHI3.5+FT(CM)	0.93	0.93
WIKIPEDIA	PHI3.5+FT(CM+HN)	0.93	0.92
WIKIPEDIA	PHI3.5+SDM+FT(CM)	0.92	0.92
WIKIPEDIA	PHI3.5+SDM+FT(CM+HN)	0.93	0.92
WIKIPEDIA	PHI3.5+SDM+FT(CM+HN)$_{\text{2nd}}$	0.93	0.93
WIKIPEDIACVS	PHI3.5	0.14	0.13
WIKIPEDIACVS	PHI3.5+FT(CM)	0.91	0.91
WIKIPEDIACVS	PHI3.5+FT(CM+HN)	0.90	0.89
WIKIPEDIACVS	PHI3.5+SDM+FT(CM)	0.89	0.89
WIKIPEDIACVS	PHI3.5+SDM+FT(CM+HN)	0.89	0.88
WIKIPEDIACVS	PHI3.5+SDM+FT(CM+HN)$_{\text{2nd}}$	0.92	0.91

Table 3: Generation results. EXACT MATCH ACCURACY only counts verbatim matches to the ground-truth (sentences, classifications, and formatting) as correct. SENTENCE ACCURACY only considers the sentence content, after parsing the first available tagged sentence.

fect of pulling closer the representations over in-distribution data. For LM applications, this enables rejecting fewer in-distribution generations, while still correctly rejecting generations unlike those seen in the SDM estimator's support set.[8]

Differences in *marginal* accuracy are likely within noise across methods in this controlled setting for these datasets, as shown in Table 3, which evaluates both the sentence content and the instruction-following quality (rightmost column) of the generations, without using the SDM layer. As observed here, **a reduction in rejections over in-distribution data is not necessarily identifiable via the marginal accuracy of the underlying model**. Points outside the HR region will tend to have lower accuracy, but also higher variance. The re-run of the full approach has a nominally higher marginal accuracy, but the size of the HR region remains relatively stable. In real-world applications where generations in the HR region are treated as automated, or semi-automated, predictions in the decision pipeline, with remaining predictions routed to alternative models or human adjudication, **fine-tuning with the SDM next-token loss is preferable given the increase of > 10 percentage points of generations in the HR region over in-distribution data**.

6 Conclusion

We introduced SDM language models, which are sequence prediction models optimized to maximize the proportion of generations in the well-calibrated, high-probability region of a final-layer SDM estimator, while maintaining the ability to reject unusual inputs. In this way, we have demonstrated a principled, data-driven approach for fine-tuning existing language model architectures as uncertainty-aware models with improved statistical efficiency compared to standard supervised fine-tuning.

[8] Undesirable behavior would be a decrease in in-distribution rejections accompanied by increased over-confidence over covariate shifted data, diminishing a key behavior of SDM estimators. For similar reasons, a simple offsetting of thresholds for a fixed α is not a viable solution for reducing abstentions in the context of SDM estimators.

Limitations

These mechanisms are of practical interest given that they can be readily added to existing pre-trained language models. Architectural changes that would require a full re-training are not needed for contemporary LMs that have already been trained using significant resources. However, for at least some applications with existing LMs, fine-tuning itself may not be necessary to achieve acceptable abstention rates and is typically not the first option to consider given the computational costs. Alternative exogenous approaches for reducing abstentions with an SDM activation layer trained over the frozen parameters of an underlying network include increasing the size of $\mathcal{D}_{\text{ca}}$ and composing over multiple models, retrieval, and/or external tools.

We hypothesize that further improving statistical efficiency (i.e., higher proportions of points in the HIGH-RELIABILITY region, ceteris paribus) may require architectural changes to existing layers. It may be effective to use an SDM activation in each layer of the network, thereby avoiding the marginalization over q and d from softmax activations from which the final-layer SDM activation must recover by estimating a re-partitioning.[9]

We hypothesize that test-time search strategies can be learned without changing the loss or basic setup beyond the data, enabling robust, uncertainty-aware conditional branching. In principle, generation can continue, or the conversation can be altogether reset with the negative sequence as part of the prompt, if a `<verified>No</verified>` control sequence is encountered. As noted in the main text, the final `Yes</verified>` or `No</verified>` tokens are intended to be force-decoded based on the SDM estimator; thus, these branching decisions would be made in the joint representation and output space of the SDM estimator, rather than the output ("token") space of typical existing test-time search strategies. We leave this to future higher-resourced experiments for examination.

Even though the CONTRASTIVE-MASKING encoding scheme masks the content of the negative sequences, if a static negative is in the far tail of the output distribution of the underlying language model π, the probability of generating the negative sequence could potentially rise (even if still low in absolute terms) relative to never fine-tuning with such a sequence. The SDM estimator constrains negative generations commensurate to the chosen α, but in the absence of a test-time SDM estimator, the test-time constraint would be the less robust estimate encoded in the final control sequences, which would be similar to using a softmax activation as an estimator of the predictive uncertainty. For this reason, we do not recommend using CONTRASTIVE-MASKING without a test-time SDM estimator.[10] Similarly, static negative examples in the tail of the output distribution of π with very high-risk content are a priori better suited for the training and calibration sets of the test-time SDM estimator trained over the frozen parameters of π.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, and 110 others. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *Preprint*, arXiv:2404.14219.

Chris Brew. 1992. Letting the cat out of the bag: generation for shake-and-bake mt. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, page 610–616, USA. Association for Computational Linguistics.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`.

[9] Separately, we hypothesize that each layer in such a network can be trained independently without back-propagation through the full network (cf., Hinton, 2022). This hypothesis follows from the fact that an SDM activation layer is itself trained by freezing the lower layers.

[10] More generally, in order to constrain hallucinations and highly-confident wrong answers, we do not recommend using any language model without at least a test-time SDM estimator of the predictive uncertainty.

Geoffrey Hinton. 2022. The forward-forward algorithm: Some preliminary investigations. *Preprint*, arXiv:2212.13345.

IEEE. 1969. Ieee recommended practice for speech quality measurements. *IEEE No 297-1969*, pages 1–24.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Allen Schmaltz. 2025. Similarity-distance-magnitude activations. *Preprint*, arXiv:2509.12760.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Appendix

A.1 Additional Implementation Details

Replication code is available at the following URL: `https://github.com/ReexpressAI/sdm_activations`

Experiment parameters. In all experiments, we fine-tune the existing parameters of PHI3.5, a Transformer-based language model (Vaswani et al., 2017), for 10 epochs[11] with an effective batch size of 64. For updating the parameters associated with the original model architecture (i.e., excluding the SDM layers), we use the AdamW optimizer (Loshchilov and Hutter, 2019) with a learning rate of 5×10^{-5} for training and a weight decay of 0.01. We use a linear learning rate schedule, increasing the learning rate from 0 to 5×10^{-5} over the first 78 steps (10% of the total 780 steps) followed by a linear decrease. These hyper-parameter settings are intended to be typical settings for supervised fine-tuning at this scale. We use the same settings across all experiments.

Generating hard negatives, if applicable, starts after completing the first epoch of fine-tuning.

The SDM activation layers use the same main settings as Schmaltz (2025). The 1-D CNN of each SDM activation layer has 1000 filters, with the input (i.e., the hidden states) mean centered using summary statistics over the given SDM layer's training set. We use a mini-batch size of 50 for training. The SDM layers are trained independently of the parameters of PHI3.5, and each of the two SDM layers (over $\mathcal{D}_{\text{tr}}$ and $\mathcal{D}_{\text{ca}}$) are trained independently. Each SDM layer is trained for 200 epochs with its own optimizer, for which we use the Adam optimizer (Kingma and Ba, 2017) (without weight decay) with a learning rate of 1×10^{-5}. The final weights of each SDM layer are chosen as those with the lowest balanced (across the two classes of the document-level classification task) average loss over the calibration set of the given SDM layer.

We evaluate over $\mathcal{D}_{\text{ca}}$ in the middle and end of each PHI3.5-parameter training epoch. As such, the SDM activation layer over $\mathcal{D}_{\text{tr}}$ is trained a total of 10 times throughout the overall fine-tuning process, and the SDM activation layer over $\mathcal{D}_{\text{ca}}$ is trained 20 times over the course of all evaluations. As noted in the main text, each SDM layer itself randomly splits $\mathcal{D}_{\text{tr}}$ xor $\mathcal{D}_{\text{ca}}$ into two approximately equal pieces. In the current experiments, the training set of each SDM layer thus consists of approximately 2,500 documents, which corresponds to $\frac{|\mathcal{D}_{\text{tr}}|}{2}$ for the SDM layer using $\mathcal{D}_{\text{tr}}$ and $\frac{|\mathcal{D}_{\text{ca}}|}{2}$ for the SDM layer using $\mathcal{D}_{\text{ca}}$, and the calibration set for each of those SDM layers consists of the remaining half of $\mathcal{D}_{\text{tr}}$ xor $\mathcal{D}_{\text{ca}}$, as applicable, again consisting of approximately 2,500 documents. This random shuffling occurs each time a given SDM layer is trained. To reduce compute time, we only perform a single iteration, $J = 1$, of 200 epochs each time a given SDM layer is trained, rather than the $J = 10$ iterations of random shuffles and parameter initializations of Schmaltz (2025). As with the settings for PHI3.5, the goal of these choices for the SDM layers is to hold the established parameter and hyper-parameter settings constant in our controlled experiments.

The experiments were implemented in PyTorch (Paszke et al., 2019) using the Hugging Face Transformers library (Wolf et al., 2020) and the Hugging Face Accelerate library for distributed training (Gugger et al., 2022). We used the GPU version of the Faiss library (Douze et al., 2024; Johnson et al., 2019) for calculating the L^2 distances of the SDM estimators. All fine-tuning runs used either 2 or 4 A100 GPUs with 80GB of memory, depending on availability, with the same effective batch size of 64.

B Additional Reference Results

For reference, Table 4 is an expanded version of Table 2 including the class- and prediction-conditional results. Table 5 contains summary statistics over the calibration splits for the SDM activation layers used at test-time.

C Notation Convention

In the present work, $\boldsymbol{z} \in \mathbb{R}^{|\mathcal{V}|}$ is the un-normalized output from the final linear layer over the vocabulary of the underlying LM. Consistent with existing works, $\boldsymbol{z}' \in \mathbb{R}^2$ is the un-normalized output from the linear layer of the SDM activation layer.

[11] Across all experiments, including the baselines, the lowest losses over $\mathcal{D}_{\text{ca}}$ occurred between half-way through all 10 epochs and prior to the final epoch.

			Class-conditional				Prediction-conditional				Marginal	
			$y = 0$		$y = 1$		$\hat{y} = 0$		$\hat{y} = 1$		$y \in \{0, 1\}$	
Dataset	Model	Estimator	Acc.	$\frac{n}{\lvert\mathcal{D}_{\text{te}}\rvert}$	Acc.	$\frac{n}{\lvert\mathcal{D}_{\text{te}}\rvert}$	Acc.	$\frac{n}{\lvert\mathcal{D}_{\text{te}}\rvert}$	Acc.	$\frac{n}{\lvert\mathcal{D}_{\text{te}}\rvert}$	Acc.	$\frac{n}{\lvert\mathcal{D}_{\text{te}}\rvert}$
Harvard	phi3.5+FT(CM)	no-reject	0.59	0.05	0.96	0.95	0.49	0.07	0.98	0.93	0.94	1.
Harvard	phi3.5+FT(CM+HN)	no-reject	0.61	0.05	0.95	0.95	0.40	0.08	0.98	0.92	0.93	1.
Harvard	phi3.5+sdm+FT(CM)	no-reject	0.50	0.07	0.96	0.93	0.46	0.07	0.96	0.93	0.93	1.
Harvard	phi3.5+sdm+FT(CM+HN)	no-reject	0.40	0.06	0.98	0.94	0.57	0.04	0.96	0.96	0.95	1.
Harvard	phi3.5+sdm+FT(CM+HN)$_{\text{2nd}}$	no-reject	0.53	0.04	0.99	0.96	0.77	0.03	0.98	0.97	0.97	1.
Wikipedia	phi3.5+FT(CM)	no-reject	0.58	0.07	0.98	0.93	0.65	0.06	0.97	0.94	0.95	1.
Wikipedia	phi3.5+FT(CM+HN)	no-reject	0.55	0.07	0.98	0.93	0.72	0.05	0.97	0.95	0.95	1.
Wikipedia	phi3.5+sdm+FT(CM)	no-reject	0.55	0.08	0.98	0.92	0.71	0.06	0.96	0.94	0.95	1.
Wikipedia	phi3.5+sdm+FT(CM+HN)	no-reject	0.54	0.07	0.99	0.93	0.75	0.05	0.97	0.95	0.95	1.
Wikipedia	phi3.5+sdm+FT(CM+HN)$_{\text{2nd}}$	no-reject	0.53	0.07	0.99	0.93	0.79	0.04	0.97	0.95	0.96	1.
WikipediaCVS	phi3.5+FT(CM)	no-reject	0.38	0.09	0.98	0.91	0.66	0.05	0.94	0.95	0.93	1.
WikipediaCVS	phi3.5+FT(CM+HN)	no-reject	0.24	0.10	1.00	0.90	0.89	0.03	0.92	0.97	0.92	1.
WikipediaCVS	phi3.5+sdm+FT(CM)	no-reject	0.27	0.11	0.99	0.89	0.73	0.04	0.92	0.96	0.91	1.
WikipediaCVS	phi3.5+sdm+FT(CM+HN)	no-reject	0.30	0.11	0.98	0.89	0.66	0.05	0.92	0.95	0.91	1.
WikipediaCVS	phi3.5+sdm+FT(CM+HN)$_{\text{2nd}}$	no-reject	0.32	0.08	0.99	0.92	0.82	0.03	0.94	0.97	0.94	1.
Harvard	phi3.5+FT(CM)	sdm$_{\text{HR}}$	0.	0.01	1.	0.46	R	0.	0.99	0.47	0.99	0.47
Harvard	phi3.5+FT(CM+HN)	sdm$_{\text{HR}}$	0.69	0.02	1.	0.34	1.	0.01	0.98	0.35	0.98	0.36
Harvard	phi3.5+sdm+FT(CM)	sdm$_{\text{HR}}$	0.36	0.02	1.	0.72	1.	0.01	0.98	0.73	0.98	0.74
Harvard	phi3.5+sdm+FT(CM+HN)	sdm$_{\text{HR}}$	0.12	0.03	1.	0.87	1.	<0.01	0.97	0.90	0.97	0.90
Harvard	phi3.5+sdm+FT(CM+HN)$_{\text{2nd}}$	sdm$_{\text{HR}}$	0.27	0.02	1.	0.88	1.	<0.01	0.99	0.89	0.99	0.90
Wikipedia	phi3.5+FT(CM)	sdm$_{\text{HR}}$	0.26	0.01	1.	0.55	1.	<0.01	0.98	0.56	0.98	0.56
Wikipedia	phi3.5+FT(CM+HN)	sdm$_{\text{HR}}$	0.22	0.02	1.	0.54	1.	<0.01	0.98	0.55	0.98	0.55
Wikipedia	phi3.5+sdm+FT(CM)	sdm$_{\text{HR}}$	0.15	0.02	1.	0.62	1.	<0.01	0.97	0.64	0.97	0.64
Wikipedia	phi3.5+sdm+FT(CM+HN)	sdm$_{\text{HR}}$	0.14	0.02	1.	0.70	1.	<0.01	0.98	0.72	0.98	0.72
Wikipedia	phi3.5+sdm+FT(CM+HN)$_{\text{2nd}}$	sdm$_{\text{HR}}$	0.17	0.02	1.	0.71	1.	<0.01	0.97	0.73	0.97	0.73
WikipediaCVS	phi3.5+FT(CM)	sdm$_{\text{HR}}$	1.	<0.01	R	0.	1.	<0.01	R	0.	1.	<0.01
WikipediaCVS	phi3.5+FT(CM+HN)	sdm$_{\text{HR}}$	0.	<0.01	1.	0.01	R	0.	0.88	0.01	0.88	0.01
WikipediaCVS	phi3.5+sdm+FT(CM)	sdm$_{\text{HR}}$	1.	<0.01	1.	<0.01	1.	<0.01	1.	<0.01	1.	<0.01
WikipediaCVS	phi3.5+sdm+FT(CM+HN)	sdm$_{\text{HR}}$	0.50	<0.01	1.	0.01	1.	<0.01	0.96	0.01	0.97	0.01
WikipediaCVS	phi3.5+sdm+FT(CM+HN)$_{\text{2nd}}$	sdm$_{\text{HR}}$	0.50	<0.01	1.	0.02	1.	<0.01	0.94	0.02	0.94	0.02

Table 4: Comparison of estimators for the word ordering task, with $\alpha = 0.95$. R indicates all predictions were rejected, which is preferred over falling under the expected accuracy. (Highlighting is applied prior to rounding.) $n = |\text{Admitted}|$, the count of non-rejected documents. Here, $\hat{y} = \arg\max \boldsymbol{z}'$, the document-level prediction from the sdm activation layer, with $y = r(\boldsymbol{s}^{+}, \hat{\boldsymbol{s}})$. For reference, we include the output from the sdm activation layer without any selective filtering as the no-reject estimator. Unlike the results on static datasets of Schmaltz (2025), in this case the true class proportions are model dependent since they are derived from test-time generations via $y = r(\boldsymbol{s}^{+}, \hat{\boldsymbol{s}})$; however, in the experiments here, the variation in proportions is well-within the resolution of the key comparisons of focus across models (namely, $\frac{n}{|\mathcal{D}_{\text{te}}|}$).

		Class-conditional								
		$y = 0$	$y = 1$	$y = 0$	$y = 1$	$y = 0$	$y = 1$	$y = 0$	$y = 1$	
sdm Calibration set	Model	sdm loss	sdm loss	Acc.	Acc.	Mean q	Mean q	ψ_0	ψ_1	q'_{min}
Wikipedia	phi3.5	0.26	0.27	0.81	0.85	218.58	36.92	∞	∞	∞
Wikipedia	phi3.5+FT(CM)	0.06	0.08	0.96	0.97	856.04	362.22	0.95	0.98	187.82
Wikipedia	phi3.5+FT(CM+HN)	0.07	0.07	0.95	0.97	862.90	240.49	0.95	0.96	139.0
Wikipedia	phi3.5+sdm+FT(CM)	0.08	0.07	0.95	0.96	682.80	386.74	0.95	0.97	162.36
Wikipedia	phi3.5+sdm+FT(CM+HN)	0.06	0.06	0.96	0.97	817.87	505.90	0.96	0.95	139.0
Wikipedia	phi3.5+sdm+FT(CM+HN)$_{\text{2nd}}$	0.06	0.05	0.96	0.98	908.75	275.94	0.96	0.95	82.0

Table 5: Summary statistics over the calibration splits of the final-layer sdm estimators used at test-time. ψ_0 and ψ_1 are the class-wise output thresholds and q'_{min} is the rescaled Similarity value used to determine the selection criteria introduced in Schmaltz (2025), which determines the high-reliability region. "sdm loss" is the loss for the binary classification task and not the next-token loss.