On Support Samples of Next Word Prediction

Yuqian Li[1]*, Yupei Du[2]*, Yufang Liu[1],
Feifei Feng[3], Mou Xiao Feng[3], Yuanbin Wu[1],
[1]School of Computer Science and Technology, East China Normal University,
[2]Utrecht University, the Netherlands,
[3]Midea Group,
liyuqian44@gmail.com, y.du@uu.nl, ybwu@cs.ecnu.edu.cn

Abstract

Language models excel in various tasks by making complex decisions, yet understanding the rationale behind these decisions remains a challenge. This paper investigates *data-centric interpretability* in language models, focusing on the next-word prediction task. Using representer theorem, we identify two types of *support samples*—those that either promote or deter specific predictions. Our findings reveal that being a support sample is an intrinsic property, predictable even before training begins. Additionally, while non-support samples are less influential in direct predictions, they play a critical role in preventing overfitting and shaping generalization and representation learning. Notably, the importance of non-support samples increases in deeper layers, suggesting their significant role in intermediate representation formation. These insights shed light on the interplay between data and model decisions, offering a new dimension to understanding language model behavior and interpretability. [1]

1 Introduction

Language models make decisions. From selecting an answer from QA benchmarks to generating reasoning steps for grade school math problems in GSM8k, language models are applauded and criticized for their decision-making ability. Like all other AI systems, understanding how decisions are made is an important research topic.

Support of decisions can be explored in different parts of language models. One common attribution is the patterns of neuron activations (*mechanistic interpretability*): some neurons are more sensitive to a specific decision and some are not. The circuit formed by active neurons mechanistically explains the model's decision (Elhage et al., 2021). Mechanistic interpretability targets on model parameters (and hidden states), however, it is not the only possible place to explore the rationale behind decisions. From the perspective of compression, parameters are the compressed version of training data. So, a natural question is whether we can trace model decisions back to their remote origin, the data.

We follow this point of view and study *data-centric interpretability* of language models. Concretely, we will focus on decisions about choosing the next word (the essential decision of language models), and try to answer the question "Which samples contribute the most when the model decides to predict a token v?". For general machine learning problems, there are two typical methods: one is based on counterfactual argument (a sample is important if the decision will change if it is removed from the training set (Koh and Liang, 2017)), another is based on representation theorems (a sample is important if it occupies a large part of prediction parameters (Yeh et al., 2018)). The first one requires computing Hessians of model parameters which could be expensive for analyzing large language models (Grosse et al., 2023b). We thus adopt the more efficient representation-based methods. Specifically,

- We illustrate the contribution of different samples to model parameters based on a simple representation theorem for the next word predictor. The results remind us of two types of *support samples* (those contribute a lot), one attracts the predictor to predict v, while another pushes the predictor away from predicting v.
- We show that, given a dataset and a model configuration, being a support sample is an intrinsic property of that sample, in the sense that 1) non-support samples have limited influence on learning support samples, and 2)

* Equal contribution.

[1]Our source code is publicly available at https://github.com/liyuqian44/On-Support-Samples-of-Next-Word-Prediction.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 10277–10289
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

it is possible to predict (with 80% accuracy) whether a sample is a support one at the very early phase of language model training, even without any training.

- We find that though non-support samples are less noticeable in the next-word predictor, it may play an important role in controlling generalization and representations learning. Specifically, we observe that 1) without any non-support samples, learning the predictor will suffer from unnecessary overfitting, and a small number of non-support samples can greatly alleviate the problem. 2) the proportion of non-support samples increases as the layer goes deeper, which means some of them are support samples of (thus contributing a lot to) intermediate representations.

2 Next Word Predictor Representation

We aim to find important samples to support a language model's next-word decisions. To measure importance, we first connect parameters (in particular, prediction heads at the last layer) and training samples with a representation theorem, then support samples are defined to be those samples contributing significantly to parameters.

Denote $(\mathbf{x}, \mathbf{y})$ to be a training sample, where $\mathbf{x} = x_1, x_2, ..., x_{t-1}$ is a sentence prefix, $\mathbf{y} = x_t \in V$ is the target token to predict given the prefix, and V is the vocabulary. $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ is a set of samples. We consider a decomposition of language models with a representation function $\phi(\mathbf{x}) \in \mathrm{R}^d$ and a token prediction function $f(\mathbf{x}) = \arg\max_v p(v|\mathbf{x})$, where $p(v|\mathbf{x}) = \frac{1}{Z} \exp \theta_v^T \phi(\mathbf{x})$. Let θ_ϕ, θ_v be the parameter of $\phi(\cdot)$ and $f(\cdot)$, $\theta_V = [\theta_{v_1}; \theta_{v_2}; \ldots; \theta_{v_{|V|}}]$ pack all predictor parameters (also called language model heads), and $\theta = [\theta_V; \theta_\phi]$ pack all parameters.

Theorem 1 (Crammer and Singer 2001; Yeh et al. 2018). *Assume θ is a stationary point of the loss function $L(\theta, D) = -\frac{1}{N}\sum_{i=1}^N \log p(\mathbf{y_i}|\mathbf{x_i}) + \lambda\|\theta\|_2^2$, then θ_v equals*

$$\theta_v = \frac{1}{2N\lambda}\sum_{i=1}^N (\mathbb{1}(\mathbf{y}_i = v) - p(v|\mathbf{x}_i))\phi(\mathbf{x}_i) \quad (1)$$

where $\mathbb{1}(\cdot)$ equals 1 when the argument is true, 0 otherwise.

The theorem gives a decomposition of the predictor θ_v using training samples $\phi(\mathbf{x})$. The coefficients of samples $\alpha_i \triangleq \mathbb{1}(\mathbf{y}_i = v) - p(v|\mathbf{x}_i)$ describe their importance in θ_v. A large α_i implies a bigger influence, and it also implies the sample is difficult to learn: a large α_i means $p(v|\mathbf{x}_i)$ is small when $\mathbf{y}_i = v$, or $p(v|\mathbf{x}_i)$ is large when $\mathbf{y}_i \neq v$, in both cases the probability of the ground truth token is small.

Definition 2. The support samples of predicting token v are defined to be $S_v = \{(\mathbf{x}_i, \mathbf{y}_i)|\tau \leq |\alpha_i|\}$. The support samples of the full language model are $S = \cup_v S_v$. The non-support samples are $\bar{S}$. [2]

We have seen that support samples are roughly hard-to-predict samples, here we can also inspect the opposite side by considering the relationship between non-support samples and easy-to-predict samples. Specifically, we consider $M = \{(\mathbf{x}, \mathbf{y})|f(\mathbf{x}) = \mathbf{y}, p(v|\mathbf{x}) \geq \gamma\}$, which are samples correctly predicted after training, and defined to be "memorized" by language models (Tirumala et al., 2022). The following fact shows that the memorized samples are a kind of non-support samples. [3]

Claim 3. If $\gamma, \tau \geq 0.5$, the non-support samples $\bar{S}$ contains M.

Proof. When $\gamma \geq 0.5$, $p(\mathbf{y}|\mathbf{x}) \geq \tau$ implies $f(\mathbf{x}) = \mathbf{y}$, and $M = \{(\mathbf{x}, \mathbf{y})|p(v|\mathbf{x}) \geq \gamma\}$.

At the same time, for any $(\mathbf{x}, \mathbf{y}) \in M$, if there exists a v such that $(\mathbf{x}, \mathbf{y}) \in S_v$,

- if $v = \mathbf{y}$, $\alpha = 1 - p(\mathbf{y}|\mathbf{x}) < 0.5 \leq \tau$,
- if $v \neq \mathbf{y}$, since $p(\mathbf{y}|\mathbf{x}) > 0.5$, $\alpha = p(v|\mathbf{x}) < 0.5 \leq \tau$.

There are contradictions $\alpha < \tau$ in both cases, thus for all v, $(\mathbf{x}, \mathbf{y}) \notin S_v$. □

3 Support Samples at First Glance

We now profile support samples. Our exemplar language model is trained from scratch with GPT-2 architecture (117M parameters, with $l = 12$ Transformer layers and hidden size $d = 768$) and wikitext-2 dataset (2.37M training samples). The vocabulary size $|V| = 50257$. We will investigate models with more parameters (345M, 774M, 1.5B) and larger training set (wikitext-103 with 117M

[2] We set hyperparameter $\tau = 0.9$ in this paper as justified in Appendix B.

[3] The original definition of memorized samples in (Tirumala et al., 2022) sets $\gamma = 0$. We can see a subtle rationale for introducing the confidence threshold γ: the claim here says that even a sample's correct label can be predicted, if the prediction is not confident enough, it could still be a support one.

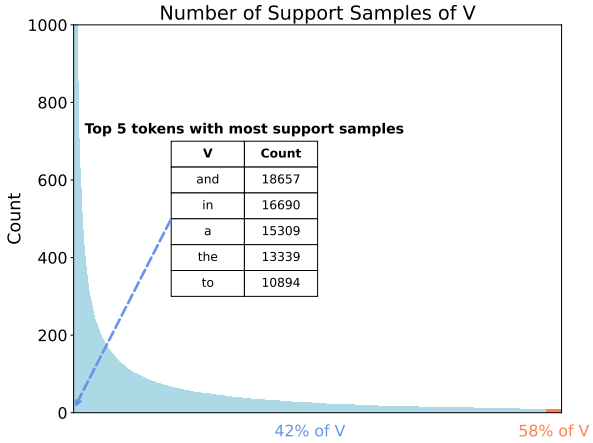


Figure 1: Number distribution of support samples for each token v. Blue bars are the top 42% tokens with the most support samples. Orange bars are the remaining 58% tokens which have less than 10 support samples. The table shows the number of the top 5 tokens.

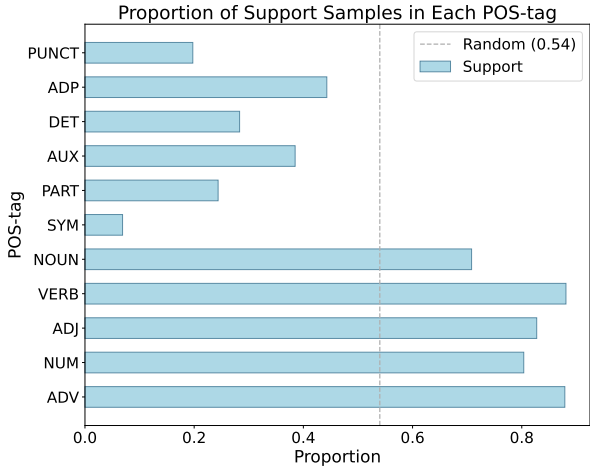


Figure 2: Proportion of support samples in different POS tags. "Random" means that a random sampling on the full dataset will pick 54% (which is the proportion of support samples) support samples. The figure shows that the proportions of support samples in two POS-tag groups significantly differ from the random proportion.

samples) in specific experiments, and report results on larger models in Appendix E.

Number of support samples. Our first observation is that, with sufficient training (loss converges) and standard model selection (best validation set performances), among 2.37M samples, there are 1.29M support samples (54%). The proportion is remarkably high: more than half of the samples are important for the next-word predictor. The larger amount of support samples implies fewer patterns are discovered during training: the predictor fails to make the correct decision by keeping a few representative samples. [4]

By depicting the number of support samples for individual $v \in V$ (Figure 1), we further find the distribution is highly screwed: very few v contribute a large number of support samples. It implies that though the overall support samples are many, prediction patterns of most tokens are clear (i.e., 58% tokens have support samples less than 10). We additionally provide a concrete example of the support and non-support samples for a specific v in Appendix C.

POS tags of support samples. To inspect support samples more closely, we roughly annotate POS tag of the next token $\mathbf{y}$ given its prefix $\mathbf{x}$. [5], and see whether samples with different target POS tags tokens will have different probabilities to be support samples. Figure 2 shows that 86% of verbs are support samples while the proportion is only 20% for punctuations. In general, support samples are more prevalent in semantically rich POS tags (e.g., ADV, NUM, ADJ, VERB, NOUN), while less common in semantically light tags (e.g., PART, AUX, DET, ADP, PUNCT). It indicates that relatively, the predictor keeps more information about verbs than punctuations in its parameters.

Two types of support samples By revisiting the representation of θ_v, we can group support samples into two types. Specifically, Equation 1 can be rewrite as (dropping $(N\lambda)^{-1}$),

$$\sum_{\mathbf{y}_i=v}(1-p(v|\mathbf{x}_i))\phi(\mathbf{x}_i)+\sum_{\mathbf{y}_i\neq v}(-p(v|\mathbf{x}_i))\phi(\mathbf{x}_i) \tag{2}$$

For each target token v, we define support samples from the first summation to be Type-1, and the second summation Type-2. Namely, Type-1 are support samples with the same target token as v, but their prediction confidence is low. Type-2 are samples with different target tokens, but they are confident in predicting the wrong target v.

To understand the roles of Type-1 and Type-2 samples, we conduct the following counterfactual argument: what if we remove (subtract) one support sample type of v from all prediction parameters θ_V. Let $(\mathbf{x}', \mathbf{y}')$ be a testing sample. Empiri-

[4] Therefore, we may also have a discuss on the definition of memorized samples: in fact, akin to the memory of a computer, we can imagine the support samples (rather than non-support samples or previously defined memorized samples (Claim 3)) are stored (or memorized) in parameters, and they will be looked up later for token prediction.

[5] https://spacy.io/

Removed Set	v="hens" full-set	v="hens" $\mathbf{y}=v$	v="ction" full-set	v="ction" $\mathbf{y}=v$
-$\emptyset$	3.28	0.24	3.28	0.56
-S_v	4.45	2.37	3.52	1.50
-Type-1 of S_v	3.35	16.73	3.28	33.09
-Type-2 of S_v	4.75	0.00	3.90	0.00
-random	3.75	0.27	3.80	0.51

Table 1: Removing support samples of v = "hens" and "ction". The number of support samples $|S_v| = 10$, and half of them are Type-1. When removing the samples, we subtract them from all θ_v. We consider four sets to remove, S_v, Type-1/Type-2 of S_v, and a random subset with the same number. We report the loss of all training samples (full-set) and loss of a subset which contains samples with v as the target token ($\mathbf{y} = v$).

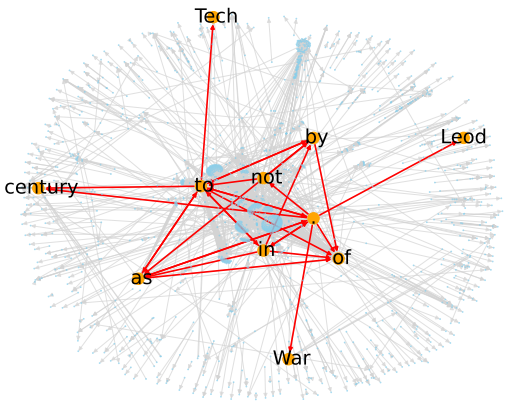
cally, we assume $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}') \geq 0$ for all i.

If we subtract Type-1 samples of v, the logit $\theta_v^T\phi(\mathbf{x}')$ will be smaller since a positive term $\sum_{\mathbf{y}=v}(1-p(\mathbf{y}_i|\mathbf{x}_i))\phi(\mathbf{x}_i)^T\phi(\mathbf{x}')$ is dropped. With a softmax activation, the probability of predicting v could be greatly decreased, and the model resists predicting v. Similarly, when subtracting Type-2 samples, a negative term is dropped, and the model tends to predict v. [6] Therefore, we would think that the existence of Type-1 support samples pull the predictor towards predicting v, while Type-2 samples push the predictor away from predicting v.

Table 1 lists a concrete example for which we conduct the above subtracting experiments. The results show that, for these two when removing Type-1, predictions on v fail, and predictions on other samples can be kept (even improved, compared with random subtracting: 3.35 vs. 3.75, and 3.28 vs. 3.80). When removing Type-2, predictions on v are perfect, while other samples are negatively affected. It shows that holding Type-2 support samples is crucial for the predictor, which is new to the perplexity-based data selection principle (only Type-1 samples are important).

To further understand Type-2, we draw a network to illustrate connections of next-token targets. In Figure 3, we link tokens according to Type-2 supporting relations: each directed edge from v to u indicates that a sample with v as the target token is the Type-2 support sample of u. We also list hubs with most in-degrees and out-degrees.

[6] We also simply (and empirically) ignore the operations' potential complex influences on other predictors (Figure 1).



v	,	@	and	.	in
out-degree	30	28	27	22	17
v	=	of	s	,	@
in-degree	242	153	144	134	84

Figure 3: An illustration of token relations according to Type-2. The number of nodes is 1690 (3% of V, not all tokens have Type-2 support samples). The number of edges is 1993. We lists tokens with top k(=5) out-degree and in-degree in the network.

4 Importance of Support Samples

Support samples contribute the most to θ_v (LM heads). This leads us to pose a question: *If the support samples are defined to be the most important samples, can we only use them (ignore those non-support samples) during training?*

We examine alternative ways of removing non-support samples: removing with sampling 1) **hard**: removing all non-support samples; 2) **random**; uniformly removing the same number of samples as the non-support; 3) **soft**: using the coefficient α_i as the probability for sampling, meaning that some non-support samples will be retained. We experiment with two training configurations: **only training LM heads** (other parameters inherited from the original trained LM) and **full model training**. The result is shown in Table 2.

For training LM heads, the answer is: Yes, we can remove non-support samples (around 40%-50% of the samples) while keeping the same performances, with weighted sampling or even random sampling. However, for training full models, the answer is No.

An interesting phenomenon is that soft sampling outperforms random sampling for training LM heads (fixed representation space), but the opposite occurs when training full model. The former tells non-support samples are not necessary for

learning heads (in the sense of soft removing). The latter implies non-support samples are important for learning representations (parts of the model beyond the heads). Otherwise, the model trained only with support samples will perform like only head tuning with support samples. A deeper exploration of this point is presented in Section 6.

Curiously, for tuning heads, hard sampling performs worse than soft sampling.(more or even exclusively support samples does not lead to better performance, and meanwhile random sampling with fewer support samples performs worse.) Observing the changes in support samples, we find that although hard sampling slightly reduces original support samples (1.29→1.18), it turns more unseen non-support samples into support(1.29→1.90). This makes the predictor's hyperplane unclearer (more support samples), and leads to a performance drop. It means without any non-support samples, learning the predictor will suffer from unnecessary overfitting to support. A small number of non-support samples can greatly alleviate the problem (soft better than hard), while support samples are still key for tuning heads (soft better than random). Regarding full model training, the lack of non-support samples poses a more significant issue, thus random sampling surpasses soft sampling, because non-support samples are more important for parts of the model beyond the heads as mentioned in the previous paragraph.

See Table 2 again, a pattern seems to emerge: the fewer the support samples, the lower the loss. We already see that the learned next-word predictor has many support samples. The next question is: *Is it possible to make the hyperplane clearer by reducing some support samples?*

We try more data, various regularization methods (L2 and dropout). Figure 4 first shows that the trends of loss and support proportion are largely consistent. And more data can help reducing support samples to make the hyperplane clearer and the predictor better, while L2 and embedding dropout fail.

In conclusion, we get 1) For tuning heads, support samples are key, but some non-support samples are also necessary. We can remove non-support samples with soft sampling. 2) Non-support samples are important for learning representations. 3) Fewer support samples indicate the clearer hyperplane and better performance of the predictor. More data vitals.

5 Predicting Support Samples

We have another discovery when training with subsets (not shown in Table 2): the loss on original support samples are almost the same for models trained with and without non-support samples (6.17 vs. 6.43). [7]

The result implies that non-support samples have limited influence on better fitting support samples, and it roughly suggests that the hardness of support samples might be "intrinsic". To make the problem clear, we cast an interesting question, *can we predict whether a sample is a support sample or not without training the language model?*

Surprisingly, we have a positive answer. We will show that given initial parameters, with simple classifiers (linear and MLP) and a limited number of training samples (annotations of support and non-support samples), one could recognize samples with high accuracy. It may confirm our "intrinsic" property intuition. The following are details.

Denote $\zeta(\mathbf{x}, \mathbf{y}, \theta^{(t)})$ to be a feature map which extracts features of a sample $(\mathbf{x}, \mathbf{y})$ according to model parameter $\theta^{(t)}$ at checkpoint t (especially, we are interested in the initial state $\theta^{(0)}$). $h_\eta(\cdot)$ is a binary classifier (with parameter η) which takes a feature vector $\mathbf{z} = \zeta(\mathbf{x}, \mathbf{y}, \theta^{(t)})$ and outputs $\mathbf{s} \in \{0, 1\}$ indicating whether the input is a support sample or not. To learn the classifier, we need a training set $E = \{(\mathbf{z}_j, \mathbf{s}_j)\}_{j=1}^K$. We experiment with the following configurations.

The feature map ζ We consider three types of features at checkpoint t,

- the last hidden vectors $\phi_{\theta^{(t)}}(\mathbf{x}) \in \mathrm{R}^d$, which is $\phi(\mathbf{x})$ evaluated with checkpoint $\theta^{(t)}$. Recalling that support samples are defined according to $p(v|\mathbf{x})$ which is a generalized linear model of $\phi(\mathbf{x})$, last hidden vectors seem to be a natural and effective choice.

- concatenation of all hidden vectors $[\phi^1_{\theta^{(t)}}(\mathbf{x}); \phi^2_{\theta^{(t)}}(\mathbf{x}); \ldots; \phi^l_{\theta^{(t)}}(\mathbf{x})] \in \mathrm{R}^{dl}$, where $\phi^j_{\theta^{(t)}}(\mathbf{x})$ is the hidden vector of layer j. The feature includes intermediate representations.

- gradient features $\nabla_\theta p(\mathbf{y}|\mathbf{x})|_{\theta^{(t)}}$. Instead of directly using the gradients, which are in the

[7] The loss of hard sampling (training only with support samples) on the original samples is 6.43 and of random sampling is 6.17. In fact, the original model is 5.30, but the comparison may not be fair since the size of the training sets are different. We thus take the random sampling 6.17 as our baseline.

Method	Test Loss	# Support Samples(M)	
		new training set (after removal)	original training set
-	5.08	1.29	1.29
hard	5.64	1.29→1.18	1.29→1.90
soft	5.13	0.87→0.87	1.29→1.33
random	5.18	0.70→0.74	1.29→1.38

(a) only training LM heads

Method	Test Loss	# Support Samples(M)	
		new training set (after removal)	original training set
-	5.08	1.29	1.29
hard	6.57	1.29→1.28	1.29→2.36
soft	5.69	0.87→1.01	1.29→1.76
random	5.47	0.70→0.73	1.29→1.45

(b) full model training

Table 2: Performance of the predictor and Change of support samples when removing non-support samples from original training set with various sampling methods. **-**: no removing (original training set). **hard**: removing all non-support samples; **random**: randomly removing the same number of samples as the non-support; **soft**: using the coefficient α_i as the probability for weighted sampling, meaning that some non-support samples will be retained. We count support samples on two types of dataset: one is new training set (after removal) as the initial number of support samples varies for each method, another is original training set.

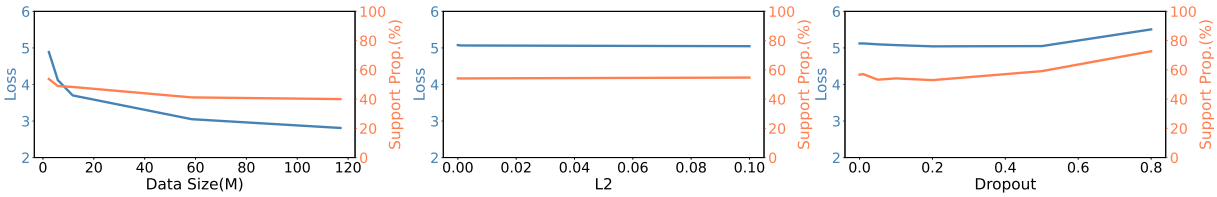


Figure 4: The change of Loss and Support proportion on different training sets. **Data Size**: we sample subsets of different sizes from wikitext-103.**L2**: we set the weight decay parameter to adjust L2 regularization during training. **Dropout**: we try different dropout rates at the embedding layer.

extremely high dimension space, we perform a random projection of the gradient vector to a low dimension (4096). [8]

Classifier We consider two classifiers,

- linear classifiers, $h_\eta(\mathbf{z}) = \eta^T \zeta(\mathbf{x}, \mathbf{y}, \theta^{(t)})$. The learnable parameter η equals the dimension of input features.
- MLPs, $h_\eta(\mathbf{z}) = w^T W_2 \sigma(W_1 \zeta(\mathbf{x}, \mathbf{y}, \theta^{(t)}))$, where σ is a non-linear activation function, and $\eta = [w; W_1; W_2]$. We try different model capacities by varying the size of W_1, W_2.

Training set We collect training set E for this classification task from the training samples of the language model D. We first train the language model with D, and get annotations of support/non-support samples according to the last checkpoint. We split the set to training, validation and testing set with 8:1:1 ratio. [9] We also test smaller versions of the training set.

Checkpoints We extract features from different checpoints of the language model. Two special checkpoints are,

- checkpoint $\theta^{(0)}$ which is the initial language model without any training.
- the last checkpoint $\theta^{(T)} = \theta$ which is the parameter defining the gold standard annotation of support/non-support samples. We may think the performances of classifiers at the last checkpoint are performances upperbounds: a classifier can see all necessary information for recognizing support samples.

[8] The time complexity of gradient features(gradients are only computed offline once) and the space complexity(gradients are projected to low dimension 4096) are acceptable.

[9] In the experiment, we use wikitext-2 set. The binary classifier has 1.90M training samples, 0.24M validation samples, and 0.24M testing samples.

As $\theta^{(0)}$ could be hard, we are also interested in the performances at the early checkpoints of training.

The experimental results (Figure 5) show that with MLP and gradient feature, the prediction accuracy can reach 80% without training, just 5% shy of the upper limit (last checkpoint). The classifier of the last checkpoint still cannot achieve 100% accuracy, suggesting that the cut between support and non-support samples is not clear. If we allow extracting features from early training phases, the prediction accuracy can be better.

Performance improves with more data and more parameters (MLP with increasing intermediate dimension) but eventually levels off. The upper limit of gradient feature is around 80%, of last hidden state is about 65%, and of all hidden states is 66%. Gradient outperforms hidden states, particularly in the early training stages. Even with more parameters, hidden states still cannot catch up.

In brief, we successfully predict whether a sample is a support sample without training (80% accuracy) only using a simple classifier (MLP) with a randomly projected gradient feature.

6 Non-support Samples

In Section 2, from the counterfactual (removing non-support samples), we preliminarily infer that non-support samples are crucial for learning model parameters beyond the heads. Now we further show that non-support samples play a role in learning representations.

In Transformers architectures, higher-layer representations are typically better than lower-layer ones. Therefore, if higher-layer representations "contain" more non-support samples, could this indicate that non-support samples contribute to learning representations? We probe each layer of LM (learning new heads for each hidden states with the original training object of the LM).

Figure 6 describes the number and POS distribution of non-support samples across different layers.[10] Overall, representations of the LM "contain" more non-support samples at higher layers. In the first six layers, the number of non-support samples remains relatively low and stable, with a sudden surge observed from the fifth to the sixth layer. The growth in higher layers is markedly more pronounced than in lower layers. For POS, the LM shows some recognition of NUM in the first layer. Notably, PUNCT is heavily contained at the sixth layer.

Now we can answer: non-support samples indeed play a role in learning representations. Conversely, they can also be viewed as a visualization of representational capacity: the LM's representational ability undergoes a qualitative leap starting at the sixth (half) layer.

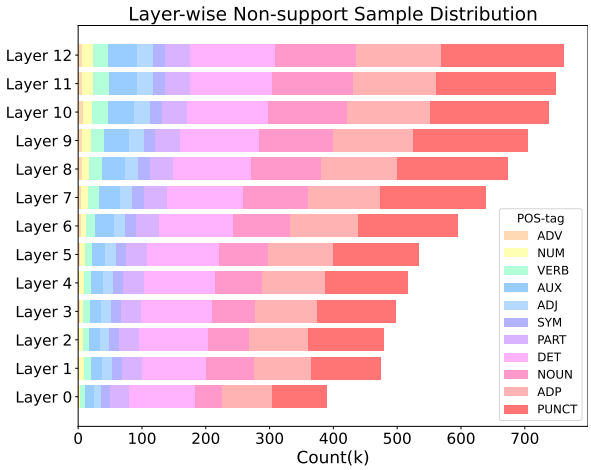


Figure 6: The number and POS distribution of non-support samples across different layers of the LM. Overall, representations of the LM "contain" more non-support samples at higher layers. In the first six layers, the number of non-support samples remains relatively low and stable, with a sudden surge observed from the fifth to the sixth layer. The growth in higher layers is markedly more pronounced than in lower layers. For POS, the LM shows some recognition of NUM in the first layer. Notably, PUNCT is heavily contained at the sixth layer.

7 Related Work

Difficult instances and model training Many studies have investigated the relationship between the difficulty of training instances and their role for model's learning, and observe that keeping harder instances and a small amount of easy instances can maintain or even improve model's generalization performance. A wide range of difficulty measures have been adopted, such as "forgetting" behavior, i.e., after which this training example is always correctly predicted (Toneva et al., 2019), training loss magnitude (Jiang et al., 2021; Swayamdipta et al., 2020), gradient norm (Paul et al., 2021), the number of layers needed for prediction (Baldock et al., 2021), perplexity (Kwok et al., 2024), and hidden representations' distance to other instances (Sorscher et al., 2022). Different from

[10] We additionally present the probing result of memorized samples in Appendix D, which as a subset of non-support samples, show similar outcomes to non-support.

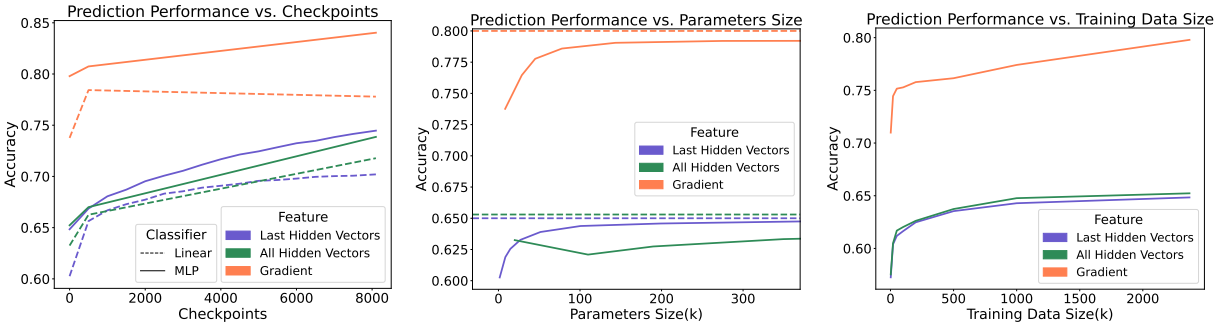


Figure 5: Performance of the classifier under various variables. **Left**: Performance changes across different training stages. The accuracy achieves nearly 80% even without training, and only improves to 85% at the last/best step. **Middle**: Performance improves with more parameters (MLP with increasing intermediate dimension) but eventually levels off. The upper limit of gradient is around 80%, of last hidden state is about 65%, and of all hidden states is 66%. Due to the varying feature dimensions (last hidden state: 768; all hidden states: 9984; gradient: projected to 4096) and significant differences in parameter counts, only the portion within 350k parameters is shown in the figure for clarity. In practice, all hidden states can achieve 66% with more parameters. **Right**: Performance enhances with the growth of training data. All three figures demonstrate that, in terms of features, gradient outperforms hidden states, particularly in the early training stages. Even with more parameters, hidden states still cannot catch up.

these works that aim to identify small but informative subsets for training, we study the specific roles of support and non-support examples in learning representations and decision boundaries.

Predictive Data Attribution Predictive data attribution seeks to answer the question: what would have happened if we had trained on a different dataset? (Madry et al., 2024). Most existing approaches rely on the gradients of individual instances.[11] One line of research uses influence functions (Koh and Liang, 2017) and its approximations (Koh et al., 2019; Park et al., 2023; Grosse et al., 2023a). However, other studies have observed them to be fragile (Basu et al., 2021) and do not answer the question of leave-one-out retraining (Bae et al., 2022). The other line of works approximates the training dynamics, by unrolling the updates during each step (Hara et al., 2019; Pruthi et al., 2020; Bae et al., 2024).

8 Conclusion

This work has taken a data-centric approach, to study the training samples that contribute the most to the prediction of each token using a representation theorem. Concretely, we make three contributions,

- We observe that support samples constitute over half of the training set. Interestingly, some predictions are supported by a large number of samples, while others are supported by only a few (§3). suggesting language models may employ two distinct modes of prediction. Furthermore, we identify two types of support samples: those that help predict the correct tokens and those that prevent incorrect token predictions.
- We further study the roles of support samples in training using removing-retraining experiments. We find that support and non-support samples are respectively more useful for learning prediction heads and backbone representations (§4 and §6). Interestingly, we observe that a few non-support samples are always needed for successful training.
- Inspired by the observation that non-support samples do not contribute to learning support samples, we investigate whether the difficulty of support samples is intrinsic by predicting them without training language models. Surprisingly, we observe that using only the gradients of a randomly initialized language model can predict support samples with over 80% accuracy, validating our hypothesis (§5).

9 Limitation

On the ambiguity between support and non-support samples. In section 6, we have noted that even when using features from the last checkpoint for training, the classifier's accuracy on the test set is still not perfect. This suggests that the

[11] A notable exception is the work of Ilyas et al. (2022), which uses a linear model of instances to predict test accuracy.

boundary between support and non-support samples is not well-defined. Directly dividing training samples into support and non-support based on a threshold may have inherent issues.

Beyond argmax: rethinking decoding and sample importance. The argmax decoding enters the representation theorem through the MLE loss function(Theorem 1), which is common in LM training. If the language model is trained in a different object function, both the representation of next-word prediction heads and the definition of important samples could be different.

However, for inference, the analysis is not limited to the argmax decoding. For example, when estimating importance scores, one could use sampling instead of argmax prediction. The sampling-based decoding may better reflect the nature of language data, which often does not align with a deterministic majority-vote interpretation. This may help address the concern that the observed next token y should not be assumed to be the argmax outcome of the underlying distribution.

In terms of the sparsity of next-word distribution and the role of argmax decoding. We would think that the learning of language model tries to establish an approximate mapping between context-dependent semantics of a token v (i.e., presentations of various prefixes ended with v) and context-independent semantics of the token (i.e., presentations of v in vocabulary). We would also think that the next-word prediction task is more like a nearest-neighbor search in the learned vocabulary space.

10 Acknowledgement

The authors wish to thank all reviewers for their helpful comments and suggestions. The corresponding authors are Yuanbin Wu and Yufang Liu.

References

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Baker Grosse. 2024. Training data attribution via approximate unrolling. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Juhan Bae, Nathan Hoyen Ng, Alston Lo, Marzyeh Ghassemi, and Roger Baker Grosse. 2022. If influence functions are the answer, then what is the question? In *Advances in Neural Information Processing Systems*.

Robert John Nicholas Baldock, Hartmut Maennel, and Behnam Neyshabur. 2021. Deep learning through the lens of example difficulty. In *Advances in Neural Information Processing Systems*.

Samyadeep Basu, Phil Pope, and Soheil Feizi. 2021. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*.

Koby Crammer and Yoram Singer. 2001. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

Roger Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamilė Lukošiūtė, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. 2023a. Studying large language model generalization with influence functions. *Preprint*, arXiv:2308.03296.

Roger B. Grosse, Juhan Bae, Cem Anil, Nelson Elhage, Alex Tamkin, Amirhossein Tajdini, Benoit Steiner, Dustin Li, Esin Durmus, Ethan Perez, Evan Hubinger, Kamile Lukosiute, Karina Nguyen, Nicholas Joseph, Sam McCandlish, Jared Kaplan, and Samuel R. Bowman. 2023b. Studying large language model generalization with influence functions. *CoRR*, abs/2308.03296.

Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. 2019. Data cleansing for models trained with sgd. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. 2022. Datamodels: Predicting predictions from training data. In *ICML*.

Ziheng Jiang, Chiyuan Zhang, Kunal Talwar, and Michael C Mozer. 2021. Characterizing structural regularities of labeled data in overparameterized models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5034–5044. PMLR.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR.

Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. 2019. On the accuracy of influence functions for measuring group effects. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Devin Kwok, Nikhil Anand, Jonathan Frankle, Gintare Karolina Dziugaite, and David Rolnick. 2024. Dataset difficulty and the role of inductive bias. *arXiv preprint arXiv:2401.01867*.

Aleksander Madry, Andrew Ilyas, Logan Engstrom, Sung Min (Sam) Park, and Kristian Georgiev. 2024. Data attribution at scale. Tutorial presented at the 41st International Conference on Machine Learning (ICML).

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. 2023. Trak: Attributing model behavior at scale. In *International Conference on Machine Learning (ICML)*.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. 2021. Deep learning on a data diet: Finding important examples early in training. In *Advances in Neural Information Processing Systems*.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pages 19920–19930. Curran Associates, Inc.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. In *Advances in Neural Information Processing Systems*.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A. Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. 2022. Memorization without overfitting: Analyzing the training dynamics of large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 2019. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

Chih-Kuan Yeh, Joon Sik Kim, Ian En-Hsu Yen, and Pradeep Ravikumar. 2018. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9311–9321.

A Proof of Theorem 1

Proof. We first compute derivatives of $\log p(\mathbf{y}|\mathbf{x})$,

$$\frac{\mathrm{d}\log p(\mathbf{y}|\mathbf{x})}{\mathrm{d}\theta_v} = \frac{\mathrm{d}\left(\theta_{\mathbf{y}}^T\phi(\mathbf{x}) - \log Z\right)}{\mathrm{d}\theta_v}.$$

For the normalizer $Z = \sum_{v'} \exp\left(\theta_{v'}^T\phi(\mathbf{x})\right)$, we have,

$$\frac{\mathrm{d}\log Z}{\mathrm{d}\theta_v} = \frac{1}{Z}\exp\left(\theta_v^T\phi(\mathbf{x})\right)\cdot\phi(\mathbf{x}) = p(v|\mathbf{x})\cdot\phi(\mathbf{x}).$$

Therefore,

$$\begin{aligned}\frac{\mathrm{d}\log p(\mathbf{y}|\mathbf{x})}{\mathrm{d}\theta_v} &= \begin{cases} (1-p(\mathbf{y}|\mathbf{x}))\,\phi(\mathbf{x}), & \mathbf{y}=v \\ -p(\mathbf{y}|\mathbf{x})\phi(\mathbf{x}), & \mathbf{y}\neq v \end{cases} \\ &= (\mathbb{1}(\mathbf{y}=v) - p(\mathbf{y}|\mathbf{x}))\,\phi(\mathbf{x}).\end{aligned}$$

Next, since θ is a stationary point, $\dfrac{\partial L(\theta, D)}{\partial\theta_v} = 0$, we have

$$-\frac{1}{N}\sum_{i=1}^{N}(\mathbb{1}(\mathbf{y}_i=v) - p(v|\mathbf{x}_i))\phi(\mathbf{x}_i) + 2\lambda\theta_v = 0$$

$$\Rightarrow \theta_v = \frac{1}{2N\lambda}\sum_{i=1}^{N}(\mathbb{1}(\mathbf{y}_i=v) - p(v|\mathbf{x}_i))\phi(\mathbf{x}_i)$$

□

B Rationale Behind the Choice of Hyperparameter τ

We set $\tau = 0.9$ empirically, based on the following rationale:

Distribution of sample importance scores α. As shown in Figure 7, we observe a drastic increase of α from 9.27% to 54.16% from $\tau = 0.8$ to $\tau = 0.9$, suggesting a natural threshold for different regions.

Ablation studies for threshold parameter τ sensitivity. We report the number of support samples with different τ as illustrated in Figure 8. The sharp decrease of the number of support samples from 0.9 to 1.0 further justifies our choice.

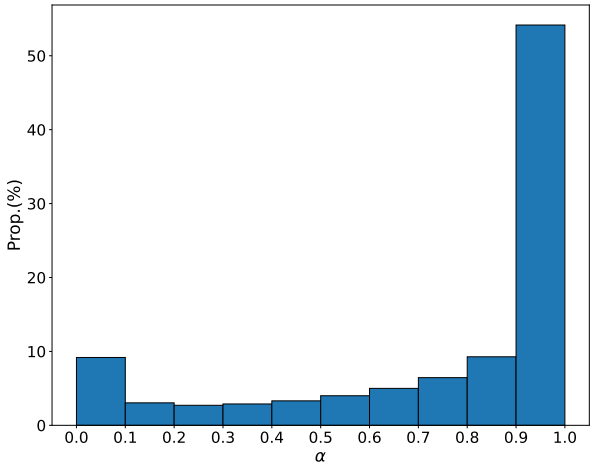


Figure 7: Distribution of sample importance scores α.

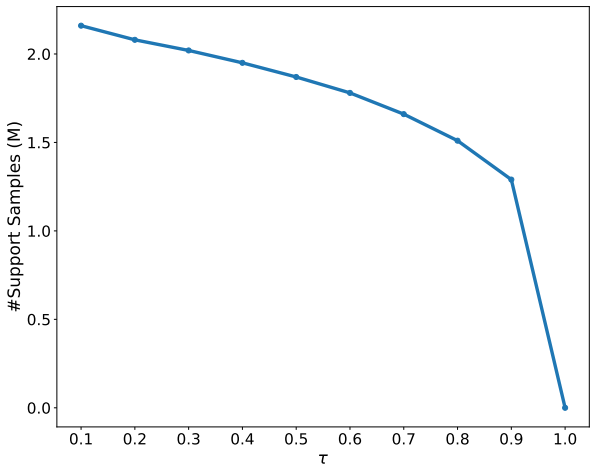


Figure 8: The number of support samples with different τ.

Softmax sharpen probabilities. Since probabilities are processed through softmax, they tend to be quite sharp. Actually, using a temperature parameter can make the model less sensitive to threshold choices, resulting in a smoother distribution.

Actually, the choice of threshold does not affect the analysis framework in the work: all methods can be generalized to any threshold.

On the other hand, threshold selection is practically significant. From a practitioner's viewpoint, with a fixed budget, determining an appropriately sized support set and exploring how to adjust thresholds for different support set sizes are valuable directions for future work.

C Concrete Example

We present some support and non-support samples of *etts*, where the target token is *etts* itself, in Table 3. In our framework, *etts* is typically a suffix observed in proper nouns such as *Burnetts*, *Corbetts*, and *Plunketts*. Our statistics reveal that *Burnetts* and *Corbetts*(of support samples) appear once each in the training data, while *Plunketts*(of non-support samples) appears 68 times. This example suggests that less frequent patterns in the training data are more likely to become support to the LM heads' parameters.

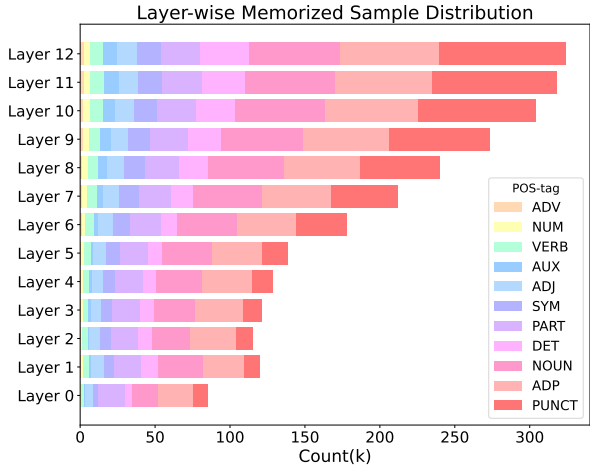


Figure 9: The number and POS distribution of memorized samples across different layers.

Type	Sample
Support	*... The northern edge of the Plunketts Creek drainage basin is formed by Burnetts*
	... They are mainly composed of granite that has weathered into more rounded hills with many long scree slopes on their flanks. The highest point of these hills is Glamaig, one of only two Corbetts
Non-support	*... Much of the Plunketts Creek valley is composed of various glacial deposits, chiefly alluvium. Although the Plunketts*

Table 3: Partial support and non-support samples of the token *etts*.

D Layer-wise Non-support/Memorized Samples Distribution

Figure 9 presents the probing result of memorized samples, which reveals a trend consistent with non-support samples as shown in Figure 6.

E Scaling Up

We scale our experiments to 345M(GPT-2-Medium architecture), 774M(GPT-2-Large architecture), and 1.5B(GPT-2-XL architecture) models. Our findings remain consistent across these scales as follows.

Method	Test Loss	# Support Samples(M)	
		new training set (after removal)	original training set
-	5.02	1.13	1.13
hard	5.62	1.13→0.98	1.13→1.77
soft	5.07	0.73→0.70	1.13→1.25
random	5.12	0.57→0.53	1.13→1.23

(a) only training LM heads

Method	Test Loss	# Support Samples(M)	
		new training set (after removal)	original training set
-	5.02	1.13	1.13
hard	6.67	1.13→1.12	1.13→2.36
soft	5.77	0.87→0.70	1.13→1.78
random	5.51	0.65→0.53	1.13→1.51

(b) full model training

Table 4: Performance of the predictor and Change of support samples of the 774M model when removing non-support samples from original training set with various sampling methods.

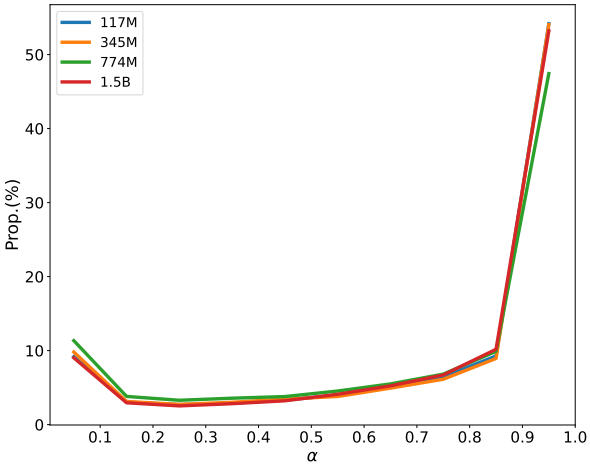


Figure 10: Distribution of samples importance scores α across models of various sizes trained on the wikitext-2 dataset.

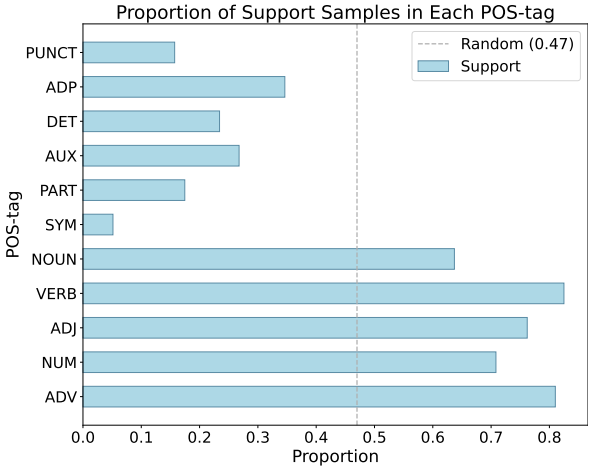


Figure 11: Proportion of support samples in different POS tags of 774M model.

Distribution of sample importance scores α. We compare the distribution across models of various sizes trained on the wikitext-2 dataset and find them similar, highlighting the scalability of our method.

Number of support samples. Table 5 lists the number of support samples for models of various sizes trained on the wikitext-2 dataset, showing consistency across scales.

We further replicate most of results using 774M model with the wikitext-2 dataset. Following are some key observations.

POS tags of support samples. Figure 11 shows that for 774M model, 82% of verbs are support samples while the proportion is only 16% for punctuations. This feature is consistent with the 117M model.

Removing non-support samples. We conduct the experiment of removing non-support samples on the 774M model. As shown in Table 4, the conclusions(§4) still hold at this larger scale. For only training LM heads, we can remove non-support samples(around 50% of the samples) while keeping the same performances, with weighted sampling. Following the same reasoning(§4), we get for tuning heads, support samples are key, but some non-support samples are also necessary. For full model training, we can't simply remove non-support samples. Soft sampling outperforms random sampling when training LM heads, but the opposite is true for full model training, suggesting non-support sam-

Model Size	Eval Loss	#Support Samples(Prop.)
117M	5.05	1.29M (54%)
345M	5.01	1.28M (54%)
774M	5.02	1.13M (47%)
1.5B	5.13	1.26M (53%)

Table 5: Evaluation loss and number of support samples for models of various sizes trained on the wikitext-2 dataset.

ples are crucial for learning representations beyond the heads.