Unveiling Decision-Making in LLMs for Text Classification : Extraction of influential and interpretable concepts with Sparse Autoencoders

Mathis Le Bail[1], **Jérémie Dentan**[1], **Davide Buscaldi**[1,2], **Sonia Vanier**[1]

[1]LIX (École Polytechnique, IP Paris, CNRS)
[2]LIPN (Sorbonne Paris Nord) **Correspondence:** mathis.le-bail@polytechnique.edu

Abstract

Sparse Autoencoders (SAEs) have been successfully used to probe Large Language Models (LLMs) and extract interpretable concepts from their internal representations. These concepts are linear combinations of neuron activations that correspond to human-interpretable features. In this paper, we investigate the effectiveness of SAE-based explainability approaches for sentence classification, a domain where such methods have not been extensively explored. We present a novel SAE-based model `ClassifSAE` tailored for text classification, leveraging a specialized classifier head and incorporating an activation rate sparsity loss. We benchmark this architecture against established methods such as ConceptShap, Independent Component Analysis, HI-Concept and a standard TopK-SAE baseline. Our evaluation covers several classification benchmarks and backbone LLMs. We further enrich our analysis with two novel metrics for measuring the precision of concept-based explanations, using an external sentence encoder. Our empirical results show that `ClassifSAE` improves both the causality and interpretability of the extracted features.

(a) Business: Airlines (b) Sci/tech: Science–Nature

(c) Business: Currency-Trade (d) Sci/tech: Processors

(e) Business: Tech–Corporations (f) Sci/tech: Cybersecurity-Spam

Figure 1: Examples of concepts discovered by `ClassifSAE` from the internals of GPT-J fine-tuned on AG News.

1 Introduction

Text classification, similar to many other NLP tasks, has seen significant performance improvements with the adoption of Large Language Models (LLMs). However, compared to more self-explainable methods, the Transformer architecture used in LLMs does not readily reveal the specific concepts it actually leverages from the text to make a labeling decision. It relies on a high-dimensional latent space where vectors lack intuitive interpretability. Concept-based interpretability methods aim to extract high-level concepts from this space (Poeta et al., 2023). The latter are directions in the latent space that seek to align with human-understandable ideas, making them more meaningful than the latent vectors themselves. In text classification, the labels may be too simplistic and various notions can result in the same decision. Therefore, extracting more nuanced concepts from the hidden states can provide deeper insights into the model's intermediate decision-making process.

To serve as a good proxy for the concepts learned by a model, the extracted directions should encompass three key properties. Effective concept vectors should be *complete* and fully reflect the internal mechanism of the inspected neural network. This means that the original layer activations can be reliably reconstructed from the concept activations. Second, the explanations provided by the activated concepts need to be *faithful* or *causal* with respect to the model's final prediction (Lyu et al., 2024). The ablation of an identified concept vector must lead to significant variations in the inferred probabilities. Finally, the directions should align closely

Findings of the Association for Computational Linguistics: EACL 2026, pages 2477–2504
March 24-29, 2026

with well-defined and semantically meaningful human notions. This is measured by the *precision* and *recall* of the concepts. A high precision ensures that the activation of a direction is reflected by the presence of a well-defined concept within the input sentence. Conversely, the recall measures how reliably the ground-truth notion activates its associated direction when present in the tested sentence. Since recall depends on ground truth labels, which are unavailable for arbitrary text, we primarily focus on the precision of the extracted concepts.

Unsupervised approaches have gained in popularity to construct interpretable linear directions from LLMs latent space. Mechanistic Interpretability is a field of research that aims to automatically break down complex neural networks into simpler and interpretable parts to gain insights into the overall system (Zhao et al., 2024a; Wang et al., 2022; Bills et al., 2023). Recent contributions in this area built on the superposition linear representation hypothesis (Elhage et al., 2022) to design scalable Sparse AutoEncoders (SAEs) for identifying meaningful directions within LLMs latent space without supervision (Cunningham et al., 2023). They are mainly trained in the broad framework of autoregressive prediction, using as input a large number of token embeddings produced by the investigated LLM. A few studies have shown the practicality of SAE features extracted from pre-trained LLMs to obtain high accuracy scores on several text classification datasets (Gao et al., 2024; Gallifant et al., 2025). However, to the best of our knowledge, in the context of text classification by language model, no prior research has thoroughly compared SAE-extracted representations and post-hoc methods from the field of concept discovery.

In this work, we address this question and we propose `ClassifSAE`, a supervised variant of the SAE to reveal sentence-level features captured from LLMs' internal representations. Our goal is to identify features that both align with interpretable concepts and strongly influence classification outcomes. From a practical standpoint, we train our SAEs from scratch on the classification dataset, as in other concept-based reference methods. This differs from previous methods that pre-train SAEs on larger datasets and select few features that are relevant for the classification task. The method requires a limited number of sentence examples, on the order of 10,000 to 100,000. This allows any user with a model fine-tuned on a given classification task to quickly capture only the concepts related to that setting. Since the expansion dimension of the SAE is larger than the number of relevant concepts for most classification tasks, we enforce the concentration of key concepts within a subset of the hidden layer by training jointly the SAE and a classifier on that subset. Additionally, to prevent the collapse of diversity into a few active features, we design a sparsity mechanism that enables better control over the maximum activation rate of the concepts. We assess representation quality using proxy metrics for completeness, causality and interpretability.

Our paper makes the following contributions:

- We propose a new supervised SAE-based method, `ClassifSAE`, to extract fine-grained concepts learned by an LLM trained for sentence classification.
- We introduce two novel metrics, `ConceptSim` and `SentenceSim`, based on an external sentence encoder to assess the precision and interpretability property of the sentence-level concepts.
- We empirically compare our extracted concepts with four baselines: TopK-SAE, ICA (Comon, 1994), ConceptSHAP (Yeh et al., 2020) and HI-Concept (Zhao et al., 2024b), across seven backbone LLMs, each fine-tuned on four distinct classification datasets.
- Using `ClassifSAE`, we obtain sparser and more monosemantic concepts while achieving second-best causality scores and requiring up to 83% less training time than HI-Concept, the state-of-the-art method, revealing a trade-off between causality and interpretability.

2 Related Work

2.1 Concepts discovery in classification

Concept-based explanations provide a more robust understanding of an LLM's internal processes compared to the often unstable token attributions from gradient-based methods (Adebayo et al., 2018). TCAV (Kim et al., 2018) was one of the first methods to identify activation-space vectors aligned with human-interpretable concepts and to quantify their influence on classification decisions. However, TCAV is supervised and requires example sets to define target concepts. Unsupervised approaches were later developed to automatically discover relevant concepts for explaining classification decisions. Yeh et al. (2020) introduced a score

to quantify the *completeness* of a concept set in reconstructing the model's original predictions. Jourdan et al. (2023) proposed COCKATIEL, a method based on Non-Negative Matrix Factorization of the activation matrix, with the factorization rank controlling the number of concepts. Recently, Zhao et al. (2024b) introduced HI-Concept, an approach which emphasizes the causal impact of extracted concepts by training an MLP with a causal loss to reconstruct the classifier's embedding space from the learned concepts. Notable progress has also been made in computer vision, where the discovery and hierarchical organization of interpretable concepts is particularly appealing due to their visual and intuitive nature (Ge et al., 2021; Fel et al., 2023; Panousis et al., 2023; Wang et al., 2024).

2.2 SAE-based concept extraction

Early work in Mechanistic Interpretability aimed to associate interpretable concepts with individual neurons (Bau et al., 2019; Dalvi et al., 2019). However, results were mixed due to *polysemanticity*: single neurons often respond to multiple unrelated features, making interpretation ambiguous (Elhage et al., 2022; Gurnee et al., 2023a). Later studies showed that linear directions in latent space tend to be more interpretable than individual neuron activations (Park et al., 2023; Marks and Tegmark, 2023; Hollinsworth et al., 2024). These approaches are supervised, requiring expert-defined concepts and example datasets, which is a limitation. This has motivated the development of unsupervised approaches to capture a broader and more diverse set of concepts. In particular, SAEs, inspired by the superposition hypothesis of linear representations (Elhage et al., 2022), have emerged as promising tools for disentangling superposed notions in the latent space of LLMs and for identifying interpretable directions (Cunningham et al., 2023).

The SAE serves as a post-hoc technique to provide interpretability for a trained LLM. The large collection of token embeddings produced by the LLM serves as input to train the SAE. The columns of the decoder matrix are interpreted as concept directions, while the corresponding activations in the hidden layer represent the strength of each concept's presence in the input. For this reason, the concepts are also referred to as SAE *features* or *directions*. Bricken et al. (2023) use an external LLM to generate explanations for each direction based on the text excerpts that most strongly activate each concept. Subsequent studies validate the interpretability of SAE-based directions in recent LLM architectures (Templeton et al., 2024; Rajamanoharan et al., 2024; Gao et al., 2024). While the original architecture enforces sparsity via an $|.|_1$ penalty, Gao et al. (2024) introduce a TopK activation function that enables direct control over $|.|_0$ sparsity by selecting a fixed number of active concepts per input. This leads to significant improvements in the sparsity–reconstruction trade-off and reduces the number of dead features at the end of training. Variants of TopK SAE have been proposed (Ayonrinde, 2024; Bussmann et al., 2024) to allow more dynamic allocation of feature capacity for tokens that are harder to reconstruct.

3 Methodology

3.1 Preliminaries

Let f be a neural network trained for text classification. Similar to Yeh et al. (2020); Zhao et al. (2024b), we define concepts as vectors in $\mathbb{R}^p$. For a layer index ℓ, let $\mathbf{h}^\ell \in \mathbb{R}^d$ denote the residual-stream representation at that layer, which captures the sequence-level information used for prediction. We aim to approximate the hidden state $\mathbf{h}^\ell$ using a combination of m sparse concept vectors. We denote by $\hat{\mathbf{h}}^\ell$ the resulting reconstruction of $\mathbf{h}^\ell$ obtained from this sparse combination.

3.2 Existing evaluation metrics

As noted in the Introduction, we evaluate three key properties of the extracted concepts: *completeness*, *causality* and *interpretability*. For the first two, we rely on established metrics from the literature. For *interpretability*, we introduce two new metrics, detailed in Section 3.3.

Evaluating *completeness* This property assesses whether the extracted concepts are sufficient to recover the model's decisions. For concepts extracted from generative models, completeness is often measured via the reconstruction error of the original hidden state. In classification settings, however, not all directions in the hidden state necessary matter for prediction, as they can be discarded by subsequent layers if not useful for the classification decision (Hernandez et al., 2024). Therefore, we use recovery accuracy (RAcc), defined in Eq. 1, as our completeness metric. RAcc is the proportion of inputs for which the model's prediction remains unchanged when the original hidden state is replaced by its concept-based reconstruction. Formally, we

decompose the prediction model into two blocks, $f = f_{\geq \ell} \circ f_{<\ell}$, where $\mathbf{h}^\ell$ denotes the output of $f_{<\ell}$. Let $\mathcal{D}$ be the dataset for the sentence classification task. High RAcc suggests that the extracted concepts retain the classification-relevant information encoded at layer ℓ, making them a reliable basis for interpreting the model's decisions.

$$\text{RAcc} = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} \mathbb{1} \begin{bmatrix} \arg\max \left(f_{\geq \ell} \left(\mathbf{h}_i^\ell \right) \right) \\ = \arg\max \left(f_{\geq \ell} \left(\hat{\mathbf{h}}_i^\ell \right) \right) \end{bmatrix} \quad (1)$$

Evaluating *causality* This property measures the influence of the extracted concepts on the model's prediction. Let $\hat{\mathbf{h}} \backslash \{j\}$ denote the reconstructed hidden state obtained by ablating the j-th concept, by setting for instance its activation to a constant value such as 0 across the dataset. Common metrics for evaluating the influence of concept j include the shift in model accuracy, the label-flip rate and the total variation distance (TVD) between the original and modified probability distributions measured after replacing $\hat{\mathbf{h}}^\ell$ with $\hat{\mathbf{h}}^\ell \backslash \{j\}$:

$$\Delta \text{Acc}_{\{j\}} = \left| \text{Acc} \left(f_{\geq \ell} \left(\hat{\mathbf{h}}^\ell \right) \right) - \text{Acc} \left(f_{\geq \ell} \left(\hat{\mathbf{h}}^\ell \backslash \{j\} \right) \right) \right| \quad (2)$$

$$\Delta f_{\{j\}} = \mathbb{1} \left[\arg\max f_{\geq \ell} \left(\hat{\mathbf{h}}^\ell \right) \neq \arg\max f_{\geq \ell} \left(\hat{\mathbf{h}}^\ell \backslash \{j\} \right) \right] \quad (3)$$

$$\text{TVD}_{\{j\}} = \frac{1}{2} \left\| f_{\geq \ell} \left(\hat{\mathbf{h}}^\ell \right) - f_{\geq \ell} \left(\hat{\mathbf{h}}^\ell \backslash \{j\} \right) \right\|_1 \quad (4)$$

Importantly, we distinguish the notions of *global* and *conditional* feature importance. Traditionally, these metrics are averaged across the entire dataset. However, this global averaging can undervalue sparse features, those that activate for only a small subset of inputs, even if they are highly influential when active. This distinction is drawn from the causal-inference literature, where it is common to report both the Average Treatment Effect (ATE), the expected change in predictions if the feature were ablated for the entire population, and the Average Treatment Effect on the Treated (ATT), which quantifies the effect only among observations where the feature is present (Morgan and Winship, 2014). Thus, we provide these metrics in the two configurations. $(\Delta Acc_{\{j\}}^{\text{global}}, \Delta f_{\{j\}}^{\text{global}}, \text{TVD}_{\{j\}}^{\text{global}})$ denotes the metrics averaged over all evaluated sentences, while $(\Delta Acc_{\{j\}}^{\text{cond}}, \Delta f_{\{j\}}^{\text{cond}}, \text{TVD}_{\{j\}}^{\text{cond}})$ only consider sentences which activate feature j.

3.3 New metrics to evaluate *Interpretability*

Existing methods for assessing interpretability either rely on human evaluators, which is not reproducible, or LLM-as-a-judge, which is sensitive to prompting (Kim et al., 2018; Bills et al., 2023; Gurnee et al., 2023b). We therefore introduce two new metrics: `ConceptSim` and `SentenceSim`. The former measures how coherent a single concept's meaning is across sentences, while the latter assesses how consistent meaning is between sentences sharing the same concepts. We evaluate the inspected LLM on a held-out test set and record concept activations per sentence. For each concept, this produces an activation vector over the test set, which we cluster in one dimension to distinguish activated from non-activated sentences. Let $\mathcal{S}^j$ be the set of sentences activating the j-th feature and N^j its cardinal. We encode each sentence using Sentence-BERT (Reimers and Gurevych, 2019). For $s_i \in \mathcal{S}^j$, let $e(s_i)$ be the sentence embedding. We evaluate the average pairwise similarity of activating sentences:

$$\texttt{ConceptSim}(j) = \frac{1}{\binom{N^j}{2}} \sum_{\substack{i,i' \in \mathcal{S}^j \\ i \neq i'}} \frac{e(s_i) \cdot e(s_{i'})}{|e(s_i)||e(s_{i'})|} \quad (5)$$

A higher value of `ConceptSim` indicates a better interpretability and monosemanticity in concept activations. If the sentences in $\mathcal{S}^j$ really share a common concept, their pairwise cosine similarity should be high. Cosine similarity has already been used by Li et al. (2024) to assess concept interpretability, showing strong correlation with human annotations. The novelty of our metric lies in computing cosine similarity after performing a one-dimensional clustering to isolate the activating sentences in $\mathcal{S}^j$. Further implementation details are provided in Appendix D, with an illustration of the pipeline in Figure 6.

We also define `SentenceSim` to assess the similarity of sentences that share the same activating concepts. Each sentence s_i is associated with the set of its p most strongly activated concepts. We measure sentence similarity based on the number of shared concepts between these sets. For an integer k, let $\texttt{SentenceSim}(s_i, k)$ denote the average cosine similarity between $e(s_i)$ and the embeddings of sentences whose concept sets share exactly k elements with that of s_i. A higher overlap in activated features should align with closer sentences in the

embedding space. To quantify this property, we average $\texttt{SentenceSim}(s_i, k)$ over all sentences to obtain $\texttt{SentenceSim}(k)$. We expect $\texttt{SentenceSim}(k)$ to increase with k, since a higher k implies greater overlap in top-p concepts.

3.4 Sparse AutoEncoders

SAEs learn a higher-dimensional sparse representation of $\mathbf{h}^\ell$. They consist of an encoder and a decoder that attempt to reconstruct the input embedding from this representation:

$$\mathbf{z} = \sigma(W^{enc}\mathbf{h} + b^{enc}) \in \mathbb{R}^m \quad (6)$$

$$\hat{\mathbf{h}} = W^{dec}\mathbf{z} + b^{dec} \in \mathbb{R}^d \quad (7)$$

where σ is an activation function. The columns of $W^{dec} \in \mathbb{R}^{d\times m}$ can be understood as the directions of the concepts extracted in the embedding space. A popular choice for σ is the TopK activation function as it enables to fix the number of non-zeros values in $\mathbf{z}$ to k per input, thereby always having the same number of allocated features per embedding to reconstruct. The training loss for TopK SAEs is often presented as:

$$\mathcal{L}^{\text{TopK}}_{\text{SAE}}(\mathbf{h}) = \frac{\|\mathbf{h} - \hat{\mathbf{h}}\|_2^2}{\|\mathbf{h} - \mathbf{h}_{\text{batch mean}}\|_2^2} + \alpha\mathcal{L}_{\text{aux}}(\mathbf{h}, \hat{\mathbf{h}}, \mathbf{z}) \quad (8)$$

where $\alpha > 0$. It is trained to minimize the reconstruction error between $\mathbf{h}$ and $\hat{\mathbf{h}}$. An auxiliary loss $\mathcal{L}_{\text{aux}}$ can be combined to mitigate the premature emergence of dead features. These are defined as the coefficients in $\mathbf{z}$ that no longer activate for any input beyond a certain training threshold. A high proportion of dead features hinders the SAE representational capacity. A popular choice for $\mathcal{L}_{\text{aux}}$ is the auxiliary loss introduced in Gao et al. (2024) inspired by the "ghost grads" method (Jermyn and Templeton, 2024).

3.5 ClassifSAE for text classification

We now introduce `ClassifSAE`, our adaptation of TopK SAE for sentence classification. In generative tasks, the SAE is typically trained on a large and diverse dataset to capture the full range of representations encoded by the model. In contrast, for text classification, we assume that a thematically focused dataset of moderate size is used to fine-tune the model. We train the SAE on this dataset to uncover only the concepts that are relevant for the classification. To capture sentence-level features, we only train the SAE on the hidden state $\mathbf{h}^\ell$ of a single token in the input sentence. For the autoregressive language models, it is the token preceding the label. For encoder-only models like BERT, it would be the `[CLS]` token. This token indeed serves as an aggregate representation of the input sentence. As the model progresses through its layers, it consolidates sentence-level information into this token, whose final representation is used for classification.

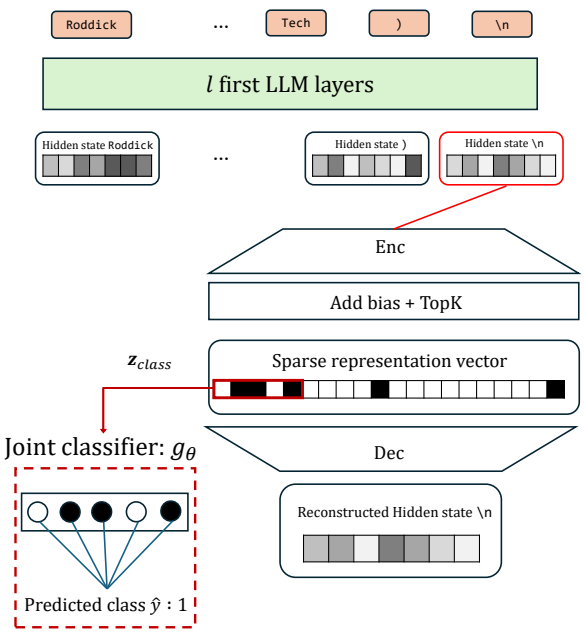


Figure 2: Architecture of our `ClassifSAE` model. A classifier is trained jointly with the SAE to replicate the original LLM prediction. A low dimensionality of $\mathbf{z}_{\text{class}}$ incentivizes the model to extract a small number of distinct task-relevant features.

Training a joint classifier We perform a forward pass of the LLM on the sentence dataset $\mathcal{D} = \{s_i\}_{i=1}^N$. This produces $X = \{(\mathbf{h}_i^\ell, \hat{y}_i)\}_{i=1}^N$, where $\hat{y}_i$ is the LLM's predicted label for sentence i. These pairs serve as inputs to jointly train the SAE and a classifier g_θ (see Figure 2). The classifier g_θ is trained to reproduce the LLM's decisions using only a subset of the SAE features $\mathbf{z}_{\text{class}} \subset \mathbf{z}$. Consequently, the SAE is incentivized to cluster task-relevant features in $\mathbf{z}_{\text{class}}$, while the remaining features support the reconstruction. We draw inspiration from Ding et al. (2020), this joint training enables end-to-end generation and selection of task-relevant features, while preserving the model's

capacity to encode task-irrelevant concepts in a distinct subspace. The classifier is trained with cross-entropy (CE) loss:

$$\mathcal{L}_{\text{class}} = \text{CE}(g_\theta(\mathbf{z}_{\text{class}}), \hat{y}) \tag{9}$$

Feature sparsity We observed that some features of the SAE are triggered by a very large portion of the dataset. Moreover, these highly-activated features exhibit a significant degree of correlation, which is not relevant for discovering diverse concepts. To alleviate these issues, we introduced an activation rate sparsity mechanism in the training phase. We select a hyperparameter γ that we consider as the targeted maximum activation rate for an individual feature. For a given batch of size B, we denote $\mathcal{I} = [\![1, B]\!]$ and $T = \lfloor \gamma B \rfloor$. We define:

$$\mathcal{I}_j = \underset{\mathcal{I}' \subseteq \mathcal{I}, |\mathcal{I}'|=T}{\arg\max} \sum_{i \in \mathcal{I}'} |z_i^j| \quad , \forall j \in [\![1, m]\!] \tag{10}$$

where z_i^j stands for the activation value of the j-th concept for input $\mathbf{h}_i$. For each feature $j \in [\![1, m]\!]$, $\mathcal{I}_j$ contains the indices of the top T inputs that most strongly activate direction j. Features with non-zero activations in more than T sentences of the batch are penalized with the following loss term:

$$\mathcal{L}_{\substack{\text{sparse} \\ \text{feature}}} = \sum_{j=1}^{m} \sum_{i' \notin \mathcal{I}_j} |z_{i'}^j| \tag{11}$$

This incentivizes a more balanced distribution of retained information across dimensions of the hidden layer, while promoting features that are more discriminative across sentences. We enforce sparsity in the activation rates of the learned directions via a penalty and not through zeroing out every exceeding activations to account for randomness in the distribution of the batch. The final training loss of `ClassifSAE` is then:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{SAE}}^{\text{TopK}} + \lambda_2 \mathcal{L}_{\text{class}} + \lambda_3 \mathcal{L}_{\substack{\text{sparse} \\ \text{feature}}} \tag{12}$$

Evaluation We retain only the task-relevant features $\mathbf{z}_{\text{class}}$ as final concepts. They are evaluated using metrics from Sections 3.2 and 3.3. For fair comparison with other methods, we zero out all z not in $\mathbf{z}_{\text{class}}$ during evaluation. The dimensionality of $\mathbf{z}_{\text{class}}$ serves as a hyperparameter controlling the number of extracted concepts. Finally, each feature is assigned the class on which it exhibits the highest average activation. See Appendix A for details.

4 Experiments and Results

We compare `ClassifSAE` to 4 concept discovery methods: ICA (Comon, 1994), TopK SAE, ConceptShap (Yeh et al. (2020)) and Hi-Concept (Zhao et al., 2024b), all trained on cached internal embeddings. The two latter and `ClassifSAE` additionally use supervision from LLM predicted labels or logits. Model descriptions and implementation details are provided in Appendix B and Appendix C, respectively. We consider four text-classification tasks: AG News (Zhang et al., 2015), IMDB (Maas et al., 2011), an offensive language identification dataset (Zampieri et al., 2019) and a sentiment analysis dataset (Rosenthal et al., 2017), both from the TweetEval benchmark (Barbieri et al., 2020). TweetEval's tasks are more challenging, with fuzzier class boundaries and skewed label distributions. We conduct experiments with seven backbone LLMs (two encoder-only and five decoder-only representatives). For the largest models, alignment with the classification task is performed via soft-prompt tuning rather than full fine-tuning. Dataset details and model training specifics are provided in Appendices E and F, respectively.

In our comparative study, concepts are extracted from the penultimate transformer block in decoder-only architectures and from the final encoder layer before the classification head in encoder-only ones, capturing high-level and sentence-aware representations. Appendix J provides an example of a more targeted analysis of layer-depth effects on concepts extraction with `ClassifSAE`. For fairness, all methods extract 20 concepts per configuration. Latent variables in $\mathbf{z}_{\text{class}}$ with near-zero mean activation are discarded. We set $\gamma = 0.1$ for sparsity and $K = 10$ for TopK activation.

4.1 Numerical results

Our results are compiled in Table 2, they are averaged over the 4 datasets. Individual results are reported in Appendix Tables 5,6, 7 and 8. Since `ConceptSim` varies significantly across datasets, we standardize the weighted average `ConceptSim` metric using the mean and variance of pairwise sentence-embedding cosine similarities within each dataset. On average, all methods maintain acceptable recovery accuracy RAcc (Eq. 1), ensuring the completeness of the learned concepts.

Concepts Interpretability `ClassifSAE` is capable of engineering features that are more interpretable according to the `ConceptSim` (Eq. 5) met-

ric. The features computed by ClassifSAE consistently exhibit higher ConceptSim scores and better activation rate sparsity across all evaluated model–task pairs. ClassifSAE architecture builds on the interpretability of the SAE and further enhances it. Because the training datasets are relatively small, the SAE revisits sentences multiple times to improve reconstruction, causing certain features to activate across a large portion of the evaluation set. This behavior is mitigated by incorporating the activation rate sparsity loss into ClassifSAE, improving sparsity and monosemanticity in the discovered concepts.

Figure 3 and 8 in Appendix show SentenceSim for the GPT-J and Pythia-1B models fine-tuned on each classification task. Sentences that share identical top-activating concepts exhibit higher similarity in the sentence embedding space when using concepts from ClassifSAE for mapping, compared to other baselines. This highlights the improved interpretability of ClassifSAE's directions.

Concepts Causality We measure causality in the *conditional* sense (see Section 3.2) so as not to artificially disadvantage sparse features. Although we define three causality metrics (ΔAcc, Δf,TVD) Eqs. 2–4, we report only Δf in Table 2, as the three are highly correlated across models and datasets and lead to the same ranking of methods. HI-Concept achieves state-of-the-art performance on this metric, which aligns with expectations since its training objective explicitly includes a surrogate for this measure. However, this addition comes at the cost of reduced interpretability: HI-Concept obtains lower average ConceptSim scores than ConceptSHAP across all models. ClassifSAE still achieves the second-best Δf^{cond} score and remains on par with HI-Concept on the largest models. Compared to standard SAE, the joint classifier improves disentanglement by promoting $\mathbf{z}_{\text{class}}$ to specialize in task-relevant concepts. Operating under a sparsity constraint and low-dimensional input, the classifier must maximize its representational capacity using a limited set of active features per sentence. This drives the selection of less correlated features in $\mathbf{z}_{\text{class}}$, improving information capture and ultimately leading to better Δf^{cond} metrics.

Computational Efficiency Table 1 reports the training times of ClassifSAE and the two most competitive baseline methods, HI-Concept and ConceptSHAP, on the AG News dataset. We use the same parameter settings as in the main comparison experiments. The post-hoc interpretability methods are branched at the output of the penultimate layer of the language model. The results show that ClassifSAE is substantially more computationally efficient, requiring up to 83% and 69% less training time than the baselines for the largest model (LLaMA 3.1 8B Instruct). This efficiency gain stems from the baselines' reliance on layers of the LLM subsequent to the one under investigation to compute reconstruction accuracy and, in the case of HI-Concept, to estimate a proxy for concept causality. As a result, their computational cost increases with both the number of downstream layers and the need to repeatedly load and process parts of the LLM. In contrast, ClassifSAE trains only an autoencoder coupled with a lightweight classifier and does not require loading any part of the LLM during training, leading to lower computational overhead. It learns a proxy of the model's behavior from the predicted labels via the joint classifier. Its cost scales only with the dimensionality of the LLM's residual stream. This computation gap further widens when concepts are extracted from earlier layers, where the baselines must process an even larger portion of the remaining model.

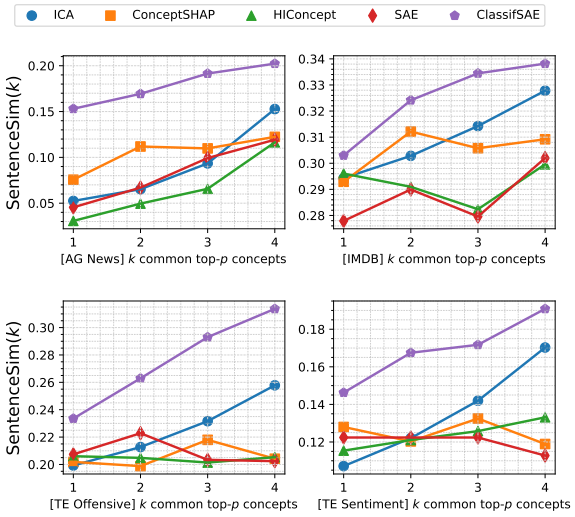


Figure 3: SentenceSim(k) as a function of the number k of shared top-activating concepts between sentence pairs. Concepts are learned from the sentence-level hidden states of the penultimate transformer block of GPT-J fine-tuned on one of the 4 classification tasks. We consider $p = 5$ concepts for each sentence.

4.2 Ablation studies

To evaluate the impact of the two newly added components in ClassifSAE and the hidden-layer size d_{sae}, we measured the completeness, causality

	Classifier Model						
	BERT(110M)	DeBERTa-v3(304M)	Pythia(410M)	Pythia(1B)	GPT-J(6B)	Mistral-Inst.(7B)	Llama-Inst.(8B)
Method							
ConceptSHAP	9 min 10 s	9 min 50 s	19 min 14 s	23 min 39 s	32 min 47 s	28 min 53 s	45 min 38 s
HI-Concept	10 min 56 s	11 min 50 s	26 min 10 s	30 min 30 s	55 min 41 s	49 min 52 s	1 h 23 min 47 s
`ClassifSAE` (K=10)	2 min 34 s	2 min 54 s	2 min 53 s	5 min 01 s	13 min 47 s	14 min 01 s	14 min 08 s

Table 1: Training time comparison between `ClassifSAE` and the two most competitive approaches (HI-Concept, ConceptSHAP) on the dataset AG News. All experiments were conducted using the same NVIDIA A100 GPU.

		Classifier Model						
		BERT(110M)	DeBERTa-v3(304M)	Pythia(410M)	Pythia(1B)	GPT-J(6B)	Mistral-Inst.(7B)	Llama-Inst.(8B)
Method	**Metric**							
ICA	RAcc (%,↑)	**99.52**	99.80	99.29	99.14	**99.50**	96.74	93.57
	Δf^{cond} (%,↑)	2.48	2.42	2.91	3.62	3.08	3.54	3.74
	Std `ConceptSim` (↑)	0.0137	0.0073	0.0175	0.0169	0.0119	0.0561	0.0443
	Avg Act. rate (%,↓)	100	100	100	100	100	100	100
ConceptSHAP	RAcc (%,↑)	96.50	98.03	95.98	94.05	96.14	93.50	89.16
	Δf^{cond} (%,↑)	6.63	0.53	4.01	3.94	2.89	3.47	2.38
	Std `ConceptSim` (↑)	0.2455	0.2135	0.1948	0.2703	0.2631	0.3694	**0.3895**
	Avg Act. rate (%,↓)	37.59	36.70	37.17	31.67	32.42	22.52	25.12
HI-Concept	RAcc (%,↑)	99.33	**99.66**	**99.40**	**99.47**	99.44	**98.98**	**98.26**
	Δf^{cond} (%,↑)	**29.66**	**33.83**	**16.81**	**16.78**	**26.19**	**16.36**	7.30
	Std `ConceptSim` (↑)	0.1799	0.2030	0.1410	0.1089	0.0624	0.0688	0.1030
	Avg Act. rate (%,↓)	39.16	41.19	46.77 5	44.38	59.11	61.83	61.33
SAE (K=10)	RAcc (%,↑)	97.39	98.47	95.54	98.08	94.46	91.27	88.90
	Δf^{cond} (%,↑)	2.04	3.43	1.98	4.28	8.97	11.60	16.03
	Std `ConceptSim` (↑)	0.3138	0.2590	0.2742	0.2207	0.1498	0.0578	0.065
	Avg Act. rate (%,↓)	20.39	29.19	18.40	20.78	35.49	44.60	41.90
`ClassifSAE` (K=10)	RAcc (%,↑)	97.745	98.35	96.99	95.94	97.06	95.93	96.35
	Δf^{cond} (%,↑)	9.86	13.96	14.48	14.92	12.03	11.30	**17.56**
	Std `ConceptSim` (↑)	**0.4512**	**0.3521**	**0.3781**	**0.3994**	**0.4657**	**0.4196**	0.3728
	Avg Act. rate (%,↓)	**9.53**	**9.38**	**8.10**	**8.48**	**9.66**	**9.85**	**10.13**

Table 2: Completeness, causality and interpretability metrics (see Sections 3.2 and 3.3) of the concepts learned from different LLM classifiers (↑ : higher is better, ↓ : lower is better). The metrics are averaged over 4 classification datasets. Prior to each evaluation, all models are fine-tuned except Mistral-Instruct and Llama-Instruct, which are aligned to the task via soft-prompt tuning (PT). Δf^{cond} (Eq. 3) is the mean of $\Delta f^{cond}_{\{j\}}$. `ConceptSim` (Eq. 5) is the weighted average of individual concept scores (Appendix D). Std `ConceptSim` denotes its standardized version. All post-hoc methods search for 20 concepts. Concepts are computed from the sentence-level hidden state, extracted for decoder-only models from the residual stream after the penultimate block, and for encoder-only models from the layer preceding the classification head.

and interpretability of the learned concepts across different training settings. Results are shown in Figure 16 in the Appendix. The SAE with both components deactivated shows the weakest `ConceptSim` and TVD^{cond} scores, irrespective of d_{sae}. The addition of the classifier for automatic variable selection significantly increases TVD^{cond}. Enabling the activation rate sparsity mechanism improves `ConceptSim` across all tested d_{sae}. Applying both strategies jointly gives the best trade-off between the evaluation metrics. While the learned concepts remain competitive for $d_{sae} \in \{512, 2048, 4096\}$ (expansion factor of 0.25, 1 and 2), TVD^{cond} shows a marked improvement with larger SAE hidden layer dimension when the joint classifier is enabled.

4.3 Concepts visualization

For each dataset, we provide a simplified 2D PCA projection of the concept embeddings learned by `ClassifSAE`. These projections are shown relative to the corresponding category prototypes in Figures 4 and 9. The visualizations also include the normalized class scores described in Appendix A. Features aligning with the same majority class naturally cluster around their prototypes, illustrating

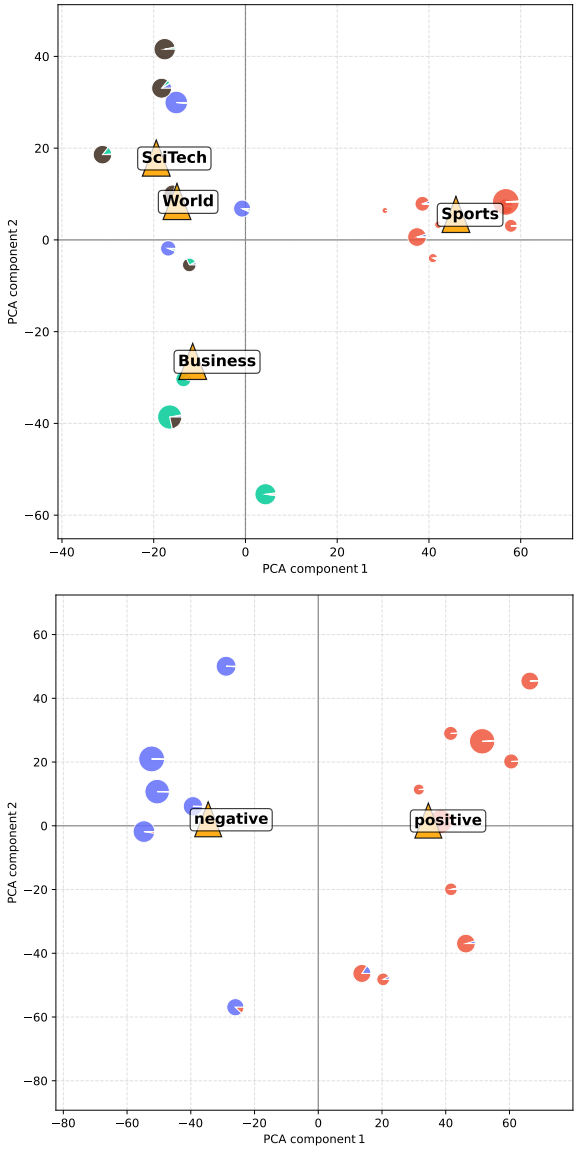


Figure 4: 2D Principal Component Analysis (PCA) fitted on sentence-level hidden-state activations extracted from the residual stream of the penultimate transformer block of LLaMA 3.1 Instruct tasked on two classification datasets: AG News (top) and IMDB (bottom). The colored circles stand for the concepts learned by `ClassifSAE`. Their size is proportional to their mean activation over the dataset. The proportion of color is representative of the normalized class score for each concept. The triangle symbols depict the class prototypes activations.

`ClassifSAE`'s ability to capture hidden-state structures that are discriminative for classification. Additionally, Figure 1 and Figures 10 and 11 in Appendix H present qualitative examples of concepts captured by `ClassifSAE` for two datasets, illustrating that the specialized classification loss does not collapse all concepts into a single one per category. It instead supports the emergence of fine-grained and semantically coherent concepts, which remain aligned with the target categories while differentiating distinct subtopics.

5 Conclusion

We introduced `ClassifSAE`, a supervised SAE-based method for discovering complete, causal and interpretable concepts from the internal representations of an LLM specialized for a text classification task. We compared this model in a comprehensive quantitative study to investigate what constitutes good concepts for explaining the decisions of black-box LLM classifiers, an area where evaluation has traditionally relied on qualitative analysis. In particular, we proposed two new metrics to quantify the monosemanticity and coherence of sentence-level features without requiring human annotations. Across multiple task–model pairs, `ClassifSAE` outperforms ICA, ConceptSHAP and TopK-SAE, showing that task-aware architecture and losses enhance SAE feature quality for the target task. While HI-Concept achieves stronger causality scores due to its explicit objective, `ClassifSAE` computes concepts up to 83% faster and surpasses it in interpretability, underscoring that current methods do not yet fully reconcile causality and interpretability and suggesting a direction for future work.

6 Limitations

While `ClassifSAE` aims to break polysemanticity in order to uncover more precise concepts, quantifying this property remains challenging, as no single metric fully captures its different aspects. We include qualitative examples to illustrate the practicality of the extracted concepts and to complement our quantitative evaluation. However, end-to-end pipelines that go beyond simple integrated gradients, linking layer-level concepts to both the input sentence and the model's output, are still sparse in the literature. Developing such pipelines could be a promising direction for future work. Our analysis also focuses on comparing concepts extracted at a fixed layer. Introducing layer choice as an additional degree of freedom could potentially yield better results, but would significantly increase computational costs, especially for large models. Finally, our study is limited to the text modality. While sparse autoencoders (SAEs) have demonstrated applicability across different modalities, investigating how our approach generalizes beyond text remains an open question.

7 Acknowledgments

This work received financial support from Crédit Agricole SA through the research chair "Trustworthy and responsible AI" with École Polytechnique. This work was granted access to the HPC resources of IDRIS under the allocation AD011015063R1 made by GENCI.

References

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity checks for saliency maps. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9525–9536, Red Hook, NY, USA. Curran Associates Inc.

Kola Ayonrinde. 2024. Adaptive sparse allocation with mutual choice & feature choice sparse autoencoders. *Preprint*, arXiv:2411.02124.

Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa-Anke, and Leonardo Neves. 2020. TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html. Accessed 2024-07-25.

Joseph Bloom, Curt Tigges, Anthony Duong, and David Chanin. 2024. Saelens. https://github.com/jbloomAus/SAELens.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nicholas L Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Chris Olah. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. https://transformer-circuits.pub/2023/monosemantic-features/index.html. Accessed: 2024-04-22.

Bart Bussmann, Patrick Leask, and Neel Nanda. 2024. Batchtopk sparse autoencoders. *Preprint*, arXiv:2412.06410.

Pierre Comon. 1994. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314. Higher Order Statistics.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. 2023. Sparse autoencoders find highly interpretable features in language models. *CoRR*, abs/2309.08600.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Zheng Ding, Yifan Xu, Weijian Xu, Gaurav Parmar, Yang Yang, Max Welling, and Zhuowen Tu. 2020. Guided variational autoencoder for disentanglement learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 7917–7926. Computer Vision Foundation / IEEE.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Preprint*, arXiv:2209.10652.

Thomas Fel, Agustin Picard, Louis Béthune, Thibaut Boissin, David Vigouroux, Julien Colin, Rémi Cadène, and Thomas Serre. 2023. Craft: Concept recursive activation factorization for explainability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2711–2721.

Jack Gallifant, Shan Chen, Kuleen Sasse, Hugo J. W. L. Aerts, Thomas Hartvigsen, and Danielle S. Bitterman.

2025. Sparse autoencoder features for classifications and transferability. *CoRR*, abs/2502.11367.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2024. Scaling and evaluating sparse autoencoders. *Preprint*, arXiv:2406.04093.

Yunhao Ge, Yao Xiao, Zhi Xu, Meng Zheng, Srikrishna Karanam, Terrence Chen, Laurent Itti, and Ziyan Wu. 2021. A Peek Into the Reasoning of Neural Networks: Interpreting with Structural Visual Concepts. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2195–2204, Nashville, TN, USA. IEEE.

Aaron Grattafiori and Abhimanyu Dubey. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023a. Finding Neurons in a Haystack: Case Studies with Sparse Probing. *Transactions on Machine Learning Research*.

Wes Gurnee, Neel Nanda, Matthew Pauly, Katherine Harvey, Dmitrii Troitskii, and Dimitris Bertsimas. 2023b. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *Preprint*, arXiv:2006.03654.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2024. Linearity of relation decoding in transformer language models. *Preprint*, arXiv:2308.09124.

Oskar John Hollinsworth, Curt Tigges, Atticus Geiger, and Neel Nanda. 2024. Language models linearly represent sentiment. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 58–87, Miami, Florida, US. Association for Computational Linguistics.

George F. Jenks. 1967. The data model concept in statistical mapping.

Adam Jermyn and Adly Templeton. 2024. Ghost grads: An improvement on resampling. https://transformer-circuits.pub/2024/jan-update/index.html. Accessed: 2025-01-10.

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.

Fanny Jourdan, Agustin Picard, Thomas Fel, Laurent Risser, Jean-Michel Loubes, and Nicholas Asher. 2023. COCKATIEL: COntinuous concept ranKed ATtribution with interpretable ELements for explaining neural net classifiers on NLP. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5120–5136, Toronto, Canada. Association for Computational Linguistics.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory sayres. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2668–2677. PMLR.

Narek Kokhlikyan, Varun Miglani, Edward Martin, Yoni Halabi Wang, Bilal Alsallakh, Jonathan Reynolds, Alex Melnikov, Nadezhda Kliushkina, Chandler Araya, Orion Yan, and David Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch. https://github.com/pytorch/captum. Accessed: 2025-06-16.

Meng Li, Haoran Jin, Ruixuan Huang, Zhihao Xu, Defu Lian, Zijia Lin, Di Zhang, and Xiting Wang. 2024. Evaluating readability and faithfulness of concept-based explanations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 607–625, Miami, Florida, USA. Association for Computational Linguistics.

Qing Lyu, Marianna Apidianaki, and Chris Callison-Burch. 2024. Towards faithful model explanation in NLP: A survey. *Computational Linguistics*, 50(2):657–723.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Samuel Marks and Max Tegmark. 2023. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *ArXiv*, abs/2310.06824.

Stephen L. Morgan and Christopher Winship. 2014. *Counterfactuals and Causal Inference: Methods and Principles for Social Research*, 2 edition. Analytical Methods for Social Research. Cambridge University Press.

Konstantinos Panousis, Dino Ienco, and Diego Marcos. 2023. Sparse Linear Concept Discovery Models. In *IEEE Xplore*, pages 2759–2763, Paris, France. IEEE, IEEE.

Kiho Park, Yo Joong Choe, and Victor Veitch. 2023. The linear representation hypothesis and the geometry of large language models. *ArXiv*, abs/2311.03658.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. 2023. Concept-based explainable artificial intelligence: A survey. *Preprint*, arXiv:2312.12936.

Senthooran Rajamanoharan, Arthur Conmy, Lewis Smith, Tom Lieberum, Vikrant Varma, János Kramár, Rohin Shah, and Neel Nanda. 2024. Improving dictionary learning with gated sparse autoencoders. *Preprint*, arXiv:2404.16014.

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.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 502–518.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Emmanuel Ameisen Craig Citro, Andy Jones, Nicholas L Turner Hoagy Cunningham, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Francesco Mosconi Tristan Hume, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Shan Carter Adam Jermyn, Chris Olah, and Tom Henighan. 2024. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html. Accessed: 2024-07-15.

Matthieu Vir. 2024. jenkspy. https://github.com/mthh/jenkspy.

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

Bor-Shiun Wang, Chien-Yi Wang, and Wei-Chen Chiu. 2024. MCPNet: An Interpretable Classifier via Multi-Level Concept Prototypes. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10885–10894, Seattle, WA, USA. IEEE.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *Preprint*, arXiv:2211.00593.

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, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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.

Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. 2020. On completeness-aware concept-based explanations in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 33, pages 20554–20565. Curran Associates, Inc.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 75–86.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA. MIT Press.

Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. 2024a. Explainability for large language models: A survey. *ACM Trans. Intell. Syst. Technol.*, 15(2).

Ruochen Zhao, Tan Wang, Yongjie Wang, and Shafiq Joty. 2024b. Explaining language model predictions with high-impact concepts. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 995–1012, St. Julian's, Malta. Association for Computational Linguistics.

A Features segmentation strategy

To enrich our framework, we rely on a segmentation scheme to cluster learned features in segments whose number matches the total available categories. The aim is to facilitate post-analysis in proposing an automatic normalized score of each class for every feature.

We denote C the set of possible categories and we remind that $\mathbf{z}$ corresponds to the projection of the sentence hidden state in the new concepts space. Therefore z^j is a scalar and the activation strength of the j-th feature. For the SAE-based methods, since we only conserve a subset of the learned latent variables, the inspected part is restricted to $\mathbf{z}_{\text{class}}$. The *mean activation* of each concept $j \in [\![1, m]\!]$ on the dataset $\mathcal{D}$ is defined as :

$$\bar{z}^j = \frac{1}{|\mathcal{D}|} \sum_{i \in \mathcal{D}} |z_i^j| \qquad , \forall j \in [\![1, m]\!] \tag{13}$$

and the *normalized score of each class* for every feature is computed as :

$$\begin{aligned} s_c(j) =& \frac{\bar{z}_c^j}{\bar{z}^j} \quad , \forall j \in [\![1, m]\!] \quad , \forall c \in C \\ =& \frac{1}{\bar{z}^j} \frac{1}{|\mathcal{D}_c|} \sum_{i \in \mathcal{D}_c} |z_i^j| \end{aligned} \tag{14}$$

where $\mathcal{D}_c$ stands for the subset of $\mathcal{D}$ which only comprises the sentences categorized as belonging to the class c. Based on these quantities, we segment features according to the class maximizing their normalized score.

$$\mathcal{F}_c := \{j : c = \arg\max_{c' \in C} s_{c'}(j)\}, \forall c \in C \tag{15}$$

We leverage the knowledge of class-specific features segments $(\mathcal{F}_c)_{c \in C}$ to account for joint global causal effect. For each label c and each ablation level $p \in \{25, 50, 70, 100\}\%$, we successively ablate the top $p\%$ features in $\mathcal{F}_c$ ranked by their mean absolute activation, leaving all other segments untouched. We then measure accuracy deterioration for each $(\mathcal{F}_c, p)$ pair and average over all segments to yield a mean impact score at ablation rate p. We perform this procedure for most of the concept-based post-hoc methods under comparison for Pythia-1B on AG News and we report $\Delta\text{Acc}^{\text{global}}$ averaged across the class-specific features segments in Figure 5. We observe that for concepts computed by `ClassifSAE`, the decline in averaged accuracy deterioration is both more pronounced and more consistent as p increases, in contrast to the other methods. Although SAE initially lags behind the others, its averaged $\Delta\text{Acc}^{\text{global}}$ converges with that of `ClassifSAE` once a whole class-specifc features segment is ablated. This pattern reflects better decorrelation among the `ClassifSAE` concepts allocated to the same class-specific segment. When features assigned to the same segment lack sufficient precision for their associated fine-grained notion, they tend to activate with similar magnitudes whenever a sentence from that class is processed, thereby negating the nuanced distinctions the concepts were meant to capture. By contrast, concepts that are less correlated and more precise each contribute incrementally to a monotonic decrease of the model's accuracy deterioration as they are sequentially removed.

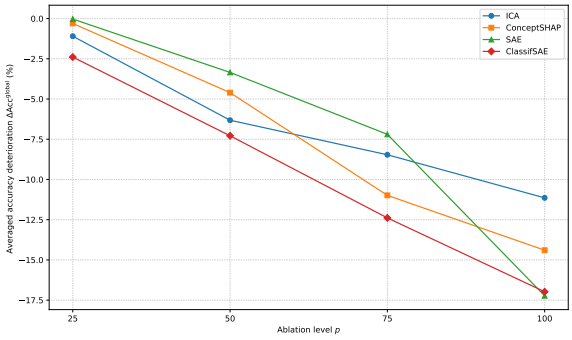


Figure 5: Averaged accuracy deterioration $\Delta\text{Acc}^{\text{global}}$ as a function of the ablation level p of class-specific features segments $(\mathcal{F}_c)_{c \in C}$. Results are reported for concepts computed from hidden states extracted at the residual stream exiting the penultimate transformer block of Pythia-1B fine-tuned on AG News.

B Baselines

Independent Component Analysis (ICA) Comon (1994) introduced this technique to separate a multivariate signal into additive, independent non-Gaussian signals, resulting in components that are more interpretable to humans than individual neurons. It is a recognized non-parametric clustering method that can be applied on neural networks activations.

ConceptShap It extracts interpretable concept vectors from neural networks by maximizing their contribution to the classifier completeness. While the term originally refers to a metric measuring the marginal contribution of each concept, it is

often used to describe the full pipeline for identifying human-aligned features proposed in Yeh et al. (2020). The semantic meaning of the concepts is emphasized by encouraging them to align closely with specific input examples. However, this clustering approach incurs a high computational cost, as it requires calculating a distance measure between all features and every sample in the training set.

HI-Concept Building on the ConceptSHAP framework, HI-Concept (Zhao et al., 2024b) introduces two innovations. It jointly minimizes a reconstruction loss to preserve the original hidden representations and explicitly maximizes the causal impact of each concept on the classifier's predictions via a causal loss that estimates treatment effects through randomized ablations of concept subsets. This causal objective, however, introduces additional computational overhead, as it requires repeated forward passes through the downstream layers of the model. As in ConceptSHAP, the semantic interpretability of concepts is encouraged through the same pair of regularization losses that promote alignment with specific input examples.

TopK SAE We compare `ClassifSAE` to TopK SAE to measure the benefits of the two additional components we introduced. The training loss is identical to that described in Section 3.1. Since there is no trained classifier clustering task-relevant features, we imitate previous approaches and train a logistic probe in post-processing to extract $\mathbf{z}_{\text{class}}$.

C Implementation details

In this section, we detail the hyperparameter settings specific to each of the compared methods. The expected number of extracted concepts is fixed and kept identical across all approaches for fair comparison. In selecting hyperparameters, our primary objective is to ensure a decent recovery accuracy, as it reflects the completeness of the learned concepts and thus their relevance as proxy of the model's internal representations. Within this constraint, we tuned parameters to jointly optimize concepts causality and interpretability, while ensuring that the recovery accuracy remained above 80%. The seed is set to 42 for all trainings.

ICA We implement it using the FastICA method from scikit-learn (Pedregosa et al., 2011), with whitening set to unit-variance, the extraction algorithm set to parallel and the maximum number of iterations fixed at 1000. Unlike the methods described below, ICA does not have a natural sparsity mechanism for the activations of learned components, such as a threshold like ConceptSHAP. Hence, the absolute values of the components' activations are never zero.

ConceptSHAP The method uses two auxiliary losses in addition of the completeness loss. Let denote $(\mathbf{c}_j)_{j\in[\![1,m]\!]}$ the concepts learned by the method. In the first regularizer term, the interpretability of each $\mathbf{c}_k$ is enhanced by maximizing $\mathbf{h}.\mathbf{c}_k$ for all sentence embeddings $\mathbf{h}$ belonging to the set of top-K nearest neighbors of $\mathbf{c}_k$. The second term enforces diversity among the learned concepts by minimizing $\mathbf{c}_k.\mathbf{c}_q$ for all pairs $(\mathbf{c}_k, \mathbf{c}_q)$. For all experiments, we set the regularizer weights to $\lambda_1 = 0.1$ and $\lambda_2 = 0.5$. The activation of $\mathbf{c}_k$ given the input $\mathbf{h}$ is computed with the formula $\text{TH}(\mathbf{h}.\mathbf{c}_k, \beta)$ where $\beta \geq 0$ acts as a threshold value below which the activation of the k-th concept is set to zero. Following guidance from (Yeh et al., 2020), we set the batch size to 128 and β is chosen within $\{0.1, 0.2, 0.3\}$. Larger values of β impose a higher activation threshold, producing sparser concepts responses and making them more likely to be interpretable, though this may come at the cost of lower recovery accuracy as some information can be discarded. Therefore, we start with $\beta = 0.3$ for each pair model-dataset and decrease its value if the recovery accuracy in the validation phase does not match the requirement. The top-K value is set to 32, a fourth of the batch size. The model is trained with an Adam optimizer and a learning rate of $3e-4$. We fixed the number of epochs at 100. The reconstruction of the original hidden state from the concepts activations is handled by a 2-layer MLP with a hidden dimension of 512.

HI-Concept For the architectural components shared with ConceptSHAP, we reuse the same hyperparameter settings, with the exception of the threshold parameter. Following the authors' recommendation, we automatically set the threshold to $\beta = \frac{1}{n}$, where n is the target number of extracted concepts. For the remaining parameters, we follow the setup of Zhao et al. (2024b): all loss components are weighted equally, and the random concept masking strategy is used for computing the causal loss. To stabilize training, the causal loss is frozen during the first half of the learning.

SAE Building on the open-source library SAELens codebase (Bloom et al., 2024) for SAE im-

plementation in language models, we adapt the method to extract concepts from a single hidden state encoding the LLM's sentence classification decision. We reuse their implementation of the ghost grad method as auxiliary loss. In all experiments, we set the training batch size to 500, we select $K = 10$ for the TopK activation function, we use the Adam optimizer with an initial learning rate of $5e-5$ and a cosine annealing schedule down to $5e-7$. For the inspected LLMs, the SAE hidden layer size $d_{sae} \in \mathbb{N}$ is set to twice the dimensionality of the input residual stream. The influence of this parameter is examined in the ablation studies in Section 4.2. As our datasets are of moderate size, the model benefits from repeated exposure to each sentence's embedding. We use a total of $10,000,000$ training tokens for all experiments. Lastly, we also take advantage of several SAE-Lens built-in utilities for encoder–decoder tied initialization, fixed-norm decoder columns, and hidden-layer activation normalization.

ClassifSAE The SAE component of the architecture is trained using the same procedure as outlined in the previous section. The main difference is the integration of $\mathcal{L}_{\text{class}}$ and $\mathcal{L}^{\text{sparse}}_{\text{feature}}$ in the training loss. We implement the joint classifier head as a single linear layer with a bias term. For deep layers, this simple architecture suffices to achieve strong reconstruction accuracy, as category-specific features are well separated in the SAE representation at that stage. We set the weights for the losses $\mathcal{L}^{\text{TopK}}_{\text{SAE}}$, $\mathcal{L}^{\text{sparse}}_{\text{feature}}$ and $\mathcal{L}_{\text{class}}$ to 0.01, 0.01 and 1, respectively.

Source Code We release the Python source code and SLURM scripts to reproduce the concepts analysis experiments presented in this paper. The repository includes the license and project documentation. The code is intended for research use only.

https://github.com/orailix/ClassifSAE

D Interpretability metrics

Sentence Encoder In all our experiments, we employ the `all-MiniLM-L6-v2` sentence-embedding model (Reimers and Gurevych, 2019) to obtain vector representations that reflect the semantic distribution of sentences in our dataset.

1D Clustering Concept activation vectors often contain small nonzero values due to noise or artifacts from SAE representations. These low-magnitude activations can blur the true firing threshold of a concept, making a fixed threshold at zero unreliable. To address this, we apply a one-dimensional clustering procedure to estimate a more semantically meaningful activation cutoff. Specifically, we use the Jenks natural breaks optimization (Jenks, 1967), which partitions the activation distribution into two clusters by minimizing intra-cluster variance. The resulting breakpoint defines the activation threshold. Sentences associated with values above it are labeled as "activating sentences" for the inspected feature. We implement this procedure using the `jenkspy` library (Vir, 2024).

Single metric for ConceptSim Since we want to evaluate the overall interpretability of the extracted concepts, we derive a single metric from the concepts scores $(\texttt{ConceptSim}(j))_{j \in [\![1,m]\!]}$, simply referred to as `ConceptSim`. Each concept $j \in [\![1, m]\!]$ is associated with a set of activating sentences and $\texttt{ConceptSim}(j)$ approximates the expected cosine similarity between two sentences randomly drawn from this set. Therefore, we report `ConceptSim` as a weighted average of the individual scores, with weights given by the number of sentence pairs $\binom{N^j}{2}$ per concept. The pair-weighted average prevents inflated scores caused by rarely active concepts that fire on a few semantically similar sentences and appear unrealistically coherent, overshadowing more frequent, low-coherence concepts

E Datasets

Dataset statistics are summarized in Table 3. AG News (Zhang et al., 2015) and IMDB (Maas et al., 2011) are class-balanced, unlike TweetEval Offensive and TweetEval Sentiment (Barbieri et al., 2020). AG News is a topic classification dataset containing news articles labeled across four categories: World, Sports, Business and Sci/Tech. IMDB is a binary sentiment analysis dataset composed of movie reviews labeled as positive or negative. TweetEval Offensive is a tweet classification dataset for detecting offensive language in social media posts and TweetEval Sentiment is a sentiment analysis dataset of tweets labeled as positive, negative or neutral.

For each task, the train split is used to train the investigated concept-based methods and we rely on the test split to report the concepts metrics.

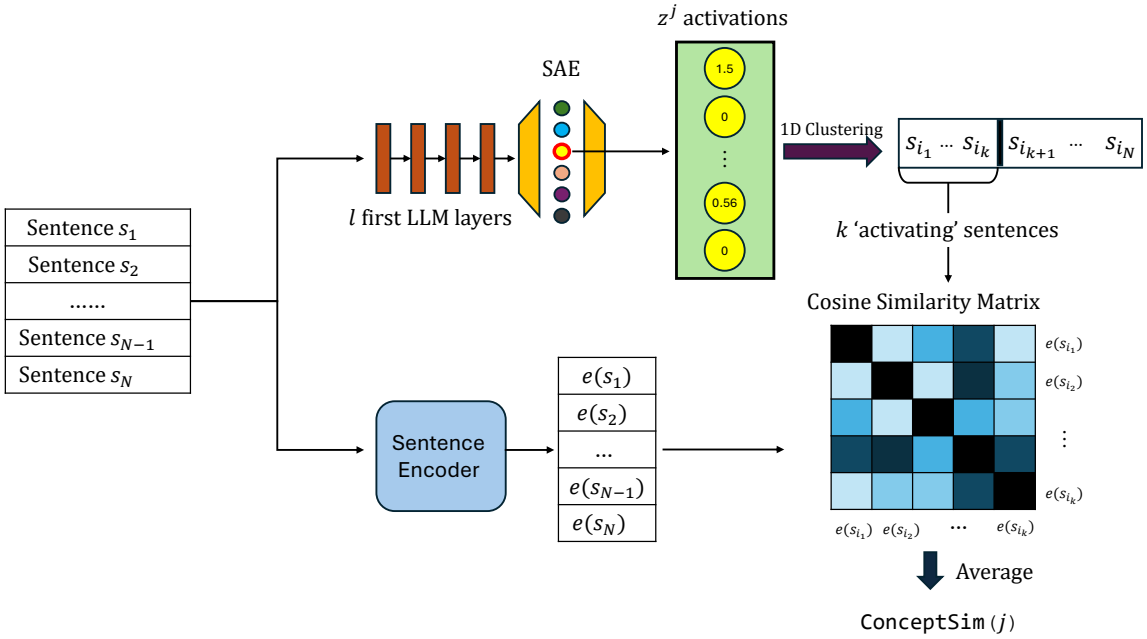


Figure 6: Illustrative explanation procedure to compute ConceptSim(j) for the j-th evaluated concept.

Dataset	Train size	Test size	Nb. labels	Avg nb. words
IMDB	25k	25k	2	234
AG News	120k	7.6k	4	38
TweetEval Offensive	12k	2k	2	23
TweetEval Sentiment	45.6k	14.3k	3	19

Table 3: Summary statistics of the datasets used in our experiments

F Classification models

To investigate models that are effective in classification settings, we fine-tune five LLM backbones independently on the four classification datasets. For each backbone–dataset pair, we train a distinct fine-tuned model. We have selected BERT-Base (Devlin et al., 2019) and DeBERTa-v3-Large (He et al., 2021) as representatives of encoder-only architectures and Pythia-410M, Pythia-1B (Biderman et al., 2023) and GPT-J-6B (Wang and Komatsuzaki, 2021) for auto-regressive architectures. Figure 7 shows an example of prompt template used to cast the auto-regressive LLMs into a classification setting by formatting each sentence accordingly. Training is performed with the `Trainer` method from HuggingFace Transformers (Wolf et al., 2020). Additionally, we include two larger models: Mistral-7B v0.1 Instruct (Jiang et al., 2023) and LLaMA 3.1 8B Instruct (Grattafiori and Dubey, 2024). They are not fine-tuned, instead we compute a task-specific prompt embedding for each architecture-dataset pair and concatenate it to the input token embeddings. This allows us to align them with the classification task without updating their weights, using soft prompt tuning. We report the individual performance of each aligned model on its corresponding dataset in Table 4.

Prompt: `Roddick and Williams to Star on Sunday at U.S. Open NEW YORK (Reuters) - The first week of the U.S. Open concludes Sunday with Andy Roddick and Serena Williams the star attractions at Flushing Meadows.\n\nOPTIONS:\n0(World)\n1(Sports)\n2(Business)\n3(Sci/Tech)\n1`

Figure 7: Example from our training set, based on the AG News dataset (Zhang et al., 2015). At the end of the sentence to be classified, the possible categories are listed along with corresponding integers, enabling the model to respond with a single token. The ground-truth label is appended at the end of the prompt: in this example, the integer 1 for the category Sports

G Computational Budget

All experiments were conducted on an HPC cluster, reaching a total of 420 hours of computation on NVIDIA A100 GPUs.

		Classifier Model			
		AG News	IMDB	TE-Off.	TE-Sent.
Method	**Metric**				
BERT	M_{train}	120000	120000	60000	180000
	Acc. (%)	93.00	93.50	82.91	70.63
	Macro-F1	0.935	0.935	0.800	0.708
DeBERTa-v3	M_{train}	120000	20000	40000	80000
	Acc. (%)	94.60	96.65	81.90	73.01
	Macro-F1	0.945	0.965	0.790	0.734
Pythia-410M	M_{train}	60000	60000	60000	80000
	Acc. (%)	94.03	95.12	80.84	71.73
	Macro-F1	0.940	0.951	0.770	0.718
Pythia-1B	M_{train}	60000	60000	60000	80000
	Acc. (%)	94.07	95.60	81.12	71.72
	Macro-F1	0.940	0.956	0.773	0.718
GPT-J	M_{train}	10000	25000	30000	24000
	Acc. (%)	93.04	96.50	81.71	73.22
	Macro-F1	0.930	0.965	0.787	0.730
Mistral-Instruct (PT)	M_{train}	4000	4000	4000	4000
	Acc. (%)	90.20	80.84	76.62	66.27
	Macro-F1	0.901	0.808	0.691	0.660
Lama-Instruct (PT)	M_{train}	4000	4000	4000	4000
	Acc. (%)	89.20	95.55	80.06	65.22
	Macro-F1	0.892	0.955	0.750	0.648

Table 4: Performance metrics of the classifiers (Accuracy and Macro-F1) across the different datasets. M_{train} denotes the total number of training sentence instances each model was exposed to during fine-tuning on the corresponding dataset. When the dataset contains fewer than M_{train} unique sentences, samples are reused across epochs and repetitions are counted towards M_{train}. For Mistral-Instruct and LLaMA-Instruct, M_{train} instead indicates the number of sentences used to compute the dataset-specific prompt embedding. Our objective is not to achieve state-of-the-art classification performance, but rather to tune models to a level of predictive reliability sufficient to enable the extraction of classification-relevant concepts.

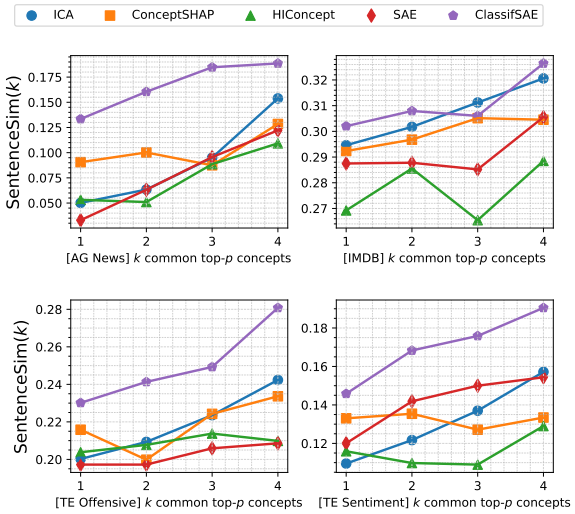


Figure 8: $\texttt{SentenceSim}(k)$ as a function of the number k of shared top-activating concepts between sentence pairs. Concepts are learned from the sentence-level hidden states of the penultimate transformer block of Pythia-1B fine-tuned on one of the 4 classification tasks. We consider $p = 5$ concepts for each sentence.

H Concepts illustrations

We provide visualizations of some concepts discovered by `ClassifSAE`, when trained on the internal activations of two distinct fine-tuned versions of GPT-J. Since displaying the top activating sentences per concept is not visually intuitive, we follow the approach of Zhao et al. (2024b) and represent concepts as word clouds. For each concept, we treat the top 500 activating sentences (or fewer, if the activating sentences set does not contain that many sentences) as a single document and compute word importance using TF-IDF. Word sizes in the resulting word cloud reflect their corresponding TF-IDF scores. For each displayed concept, we include two defining keywords in the caption. These keywords are generated by GPT-4, which is prompted with the top 20 activating sentences associated with the concept. We also display the category that each concept is predominantly associated with, based on our feature segmentation strategy detailed in Section A. For the two inspected dataset, AG News and TweetEval Sentiment, we observe that `ClassifSAE` is capable of identifying finer-grained concepts beyond the coarse label categories. Moreover, features associated with the same majority class often exhibit nuanced preferences for distinct subtopics. This illustrates that the additional loss components introduced in `ClassifSAE` do not hinder the SAE's ability to capture fine-grained and semantically precise representations

I Inputs interpretability

To illustrate one practicality of the computed concepts, we provide an example centered on input explainability. For each sentence, we first identify the neurons in $\mathbf{z}_{\text{class}}$ that are activated. Then, for each such feature, we compute token-level attributions with respect to its activation. This procedure reveals which words contribute to the capture of the associated concepts. In practice, we use the Integrated Gradients method as implemented in the Captum library (Kokhlikyan et al., 2020). Since Captum does not natively support attribution at the token level, we compute attributions with respect to

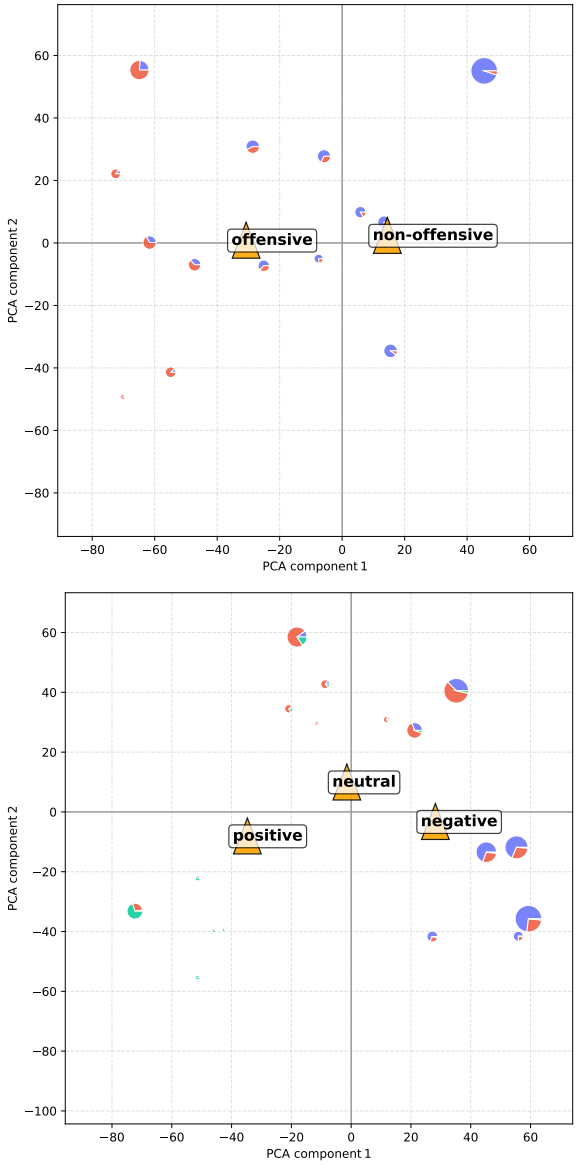


Figure 9: 2D Principal Component Analysis (PCA) fitted on sentence-level hidden-state activations extracted from the residual stream of the penultimate transformer block of LLaMA 3.1 Instruct tasked on two classification datasets: TweetEval Offensive (top) and TweetEval Sentiment (bottom). Depending from which dataset the sentence to classify come from, a prompt embedding previously computed is appended at the beginning of the sentence to align the model with the task. The colored circles stand for the concepts learned by `ClassifSAE`. Their size is proportional to their mean activation over the dataset. The proportion of color is representative of the normalized class score for each concept. The triangle symbols depict the class prototypes activations.

the token embeddings and then sum the attributions across embedding dimensions. Although gradient-based methods are noisy and often yield diffuse attributions, they nevertheless provide useful intuition about which parts of a sentence contribute most to the emergence of a given concept. Figure 12 and Figure 13 compare the attributions of concepts found by the interpretability methods HI-Concept and `ClassifSAE`. The inspected sentence was labeled as Sport in the AG News dataset, but our fine-tuned Pythia-1B model classified it under the World category. Analyzing the attributions from the 3 concepts activated by `ClassifSAE`, we observe that, despite two features associated with the Sport class being activated, the presence of the pattern '(AFP) AFP' strongly triggered a concept linked to World events, likely due to its frequent occurrence in that context. The high activation of this concept ultimately led the model to predict the World category. In contrast, the concept attributions derived from the HI-Concept method did not provide meaningful insight for this example. Furthermore, the word clouds of the activated HI-Concept features appear less precise, offering less interpretable fine-grained concepts.

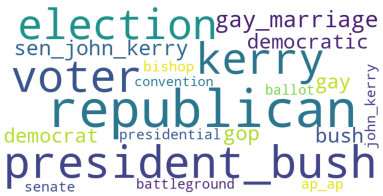
J Depth Effect on Concept Extraction

We analyze how the layer depth at which concepts are extracted by `ClassifSAE` affects the metrics of recovery accuracy (RAcc) and weighted average `ConceptSim`. Figures 14 and 15 report these metrics across layers for two configurations: Pythia-1B fine-tuned on AG News and GPT-J fine-tuned on TweetEval Offensive. In each case, the method was trained and evaluated independently at every layer to isolate the representational contribution of depth. Across both settings, we observe a steady decline in recovery accuracy toward earlier layers, with a small dip though in the mid-layers for GPT-J before improvement in the deeper ones. This trend is expected, since the representations in lower layers have not yet integrated complete sentence-level information and the embedding space at these stages is not yet specialized for the downstream classification task. Consequently, our jointly trained classifier has greater difficulty identifying discriminative sparse features to encode in $\mathbf{z}_{\text{class}}$.

For Pythia-1B tuned on AG News, the `ConceptSim` values show a more irregular trend but generally improve in the upper layers, indicating that extracted concepts become more coherent and semantically consistent as abstraction deepens. The slight rise observed in the earliest layers may correspond to stable lexical or syntactic regularities that emerge before semantic specialization. For GPT-J

tuned on TweetEval Offensive, the ConceptSim trajectory shows greater variability, with local peaks at intermediate depths. These peaks may correspond to a representational transition stage where the model organizes information into semantically coherent concept spaces that are not yet tightly aligned with the output representation, accounting for the temporary decrease in recovery accuracy at similar depths. Overall though, ConceptSim values also improve in the upper layers.

(a) World
Conflict – Middle East



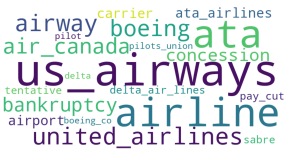
(b) World
Election - Social issues

(c) World
Asia – Tensions

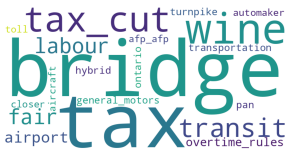
(d) Sport
College – American Football

(e) Sport
Football – Europe

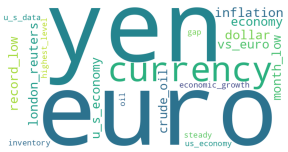
(f) Sport
Tennis - Basketball



(g) Business
Airlines - bankruptcy

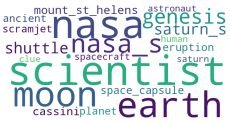


(h) Business
Infrastructure – Policy



(i) Business
Currency - Trade

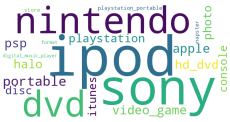
(j) Business
Tech – Corporations



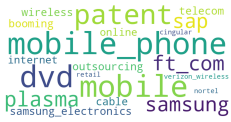
(k) Sci/tech
Science – Nature

(l) Sci/tech
Processors – Servers

(m) Sci/tech
Cybersecurity - Spam

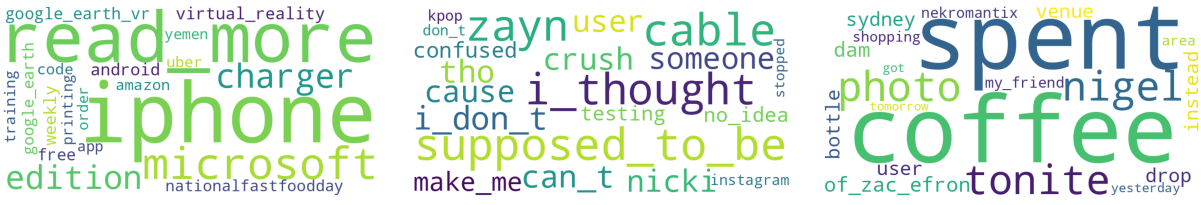


(n) Sci/tech
Gadgets – Gaming



(o) Sci/tech
Mobile - Telecom

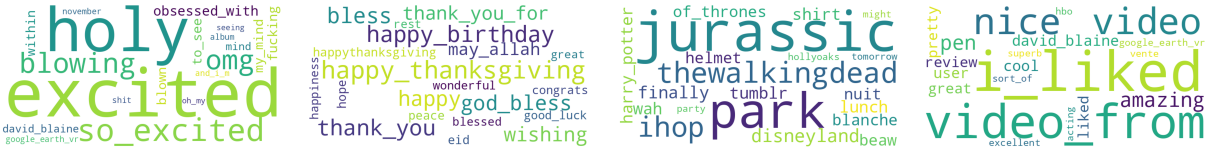
Figure 10: Examples of concepts discovered by `ClassifSAE` from the internals of GPT-J fine-tuned on AG News.



(a) Neutral
Tech - Innovation

(b) Neutral
Confused - Unsure

(c) World
Moments – Daily life



(d) Positive
Awe – Obsession

(e) Positive
Blessings – Gratitude

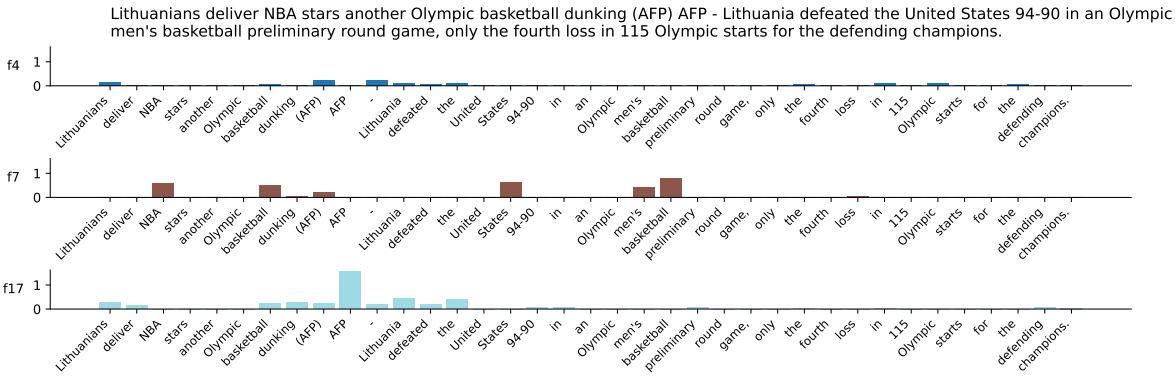
(f) Positive
Fandom – Anticipation

(g) Positive
Reaction – Content

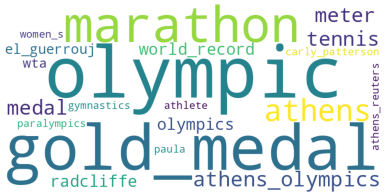
(h) Negative
Extremism – Ideology

(i) Negative
War – Crisis

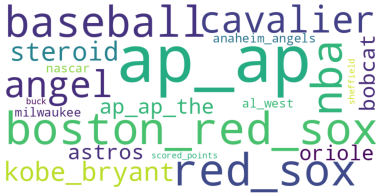
Figure 11: Examples of concepts discovered by `ClassifSAE` from the internals of GPT-J fine-tuned on TweetEval Sentiment



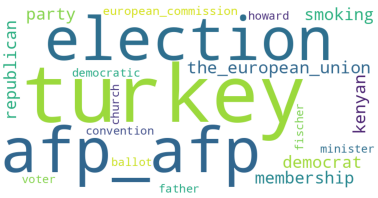
(a) Attribution of each word with regard to the activation of the 3 active feature in $\mathbf{z}_{\text{class}}$ for this sentence. Attributions are computed via Integrated Gradients method as implemented in the Captum library (Kokhlikyan et al., 2020)



(b) Feature 4 - Category Sport Concept: Olympic

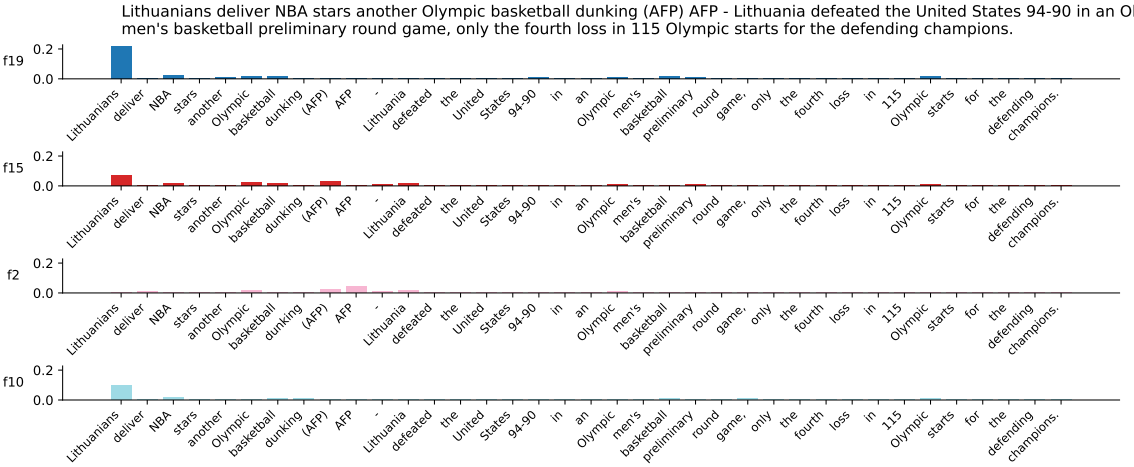


(c) Feature 7 - Category Sport Concept: American Sport



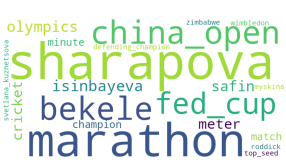
(d) Feature 17 - Category World Concept: AFP

Figure 12: Example of a sentence misclassified by our fine-tuned Pythia-1B model on the AG News dataset. The true label is Sport, but the model predicted the World category. The activated concepts computed by `ClassifSAE` are shown, along with their respective attributions over the words in the sentence



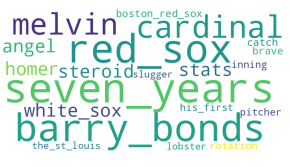
(a) Attribution of each word with regard to the activation of the top 4 most active features in $\mathbf{z}_{\text{class}}$ for this sentence. Attributions are computed via Integrated Gradients method as implemented in the Captum library (Kokhlikyan et al., 2020).

(b) Feature 19 - Category Sport



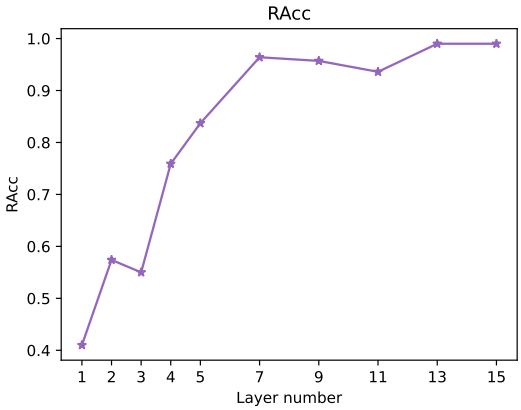
(c) Feature 15 - Category Sport

(d) Feature 2 - Category World

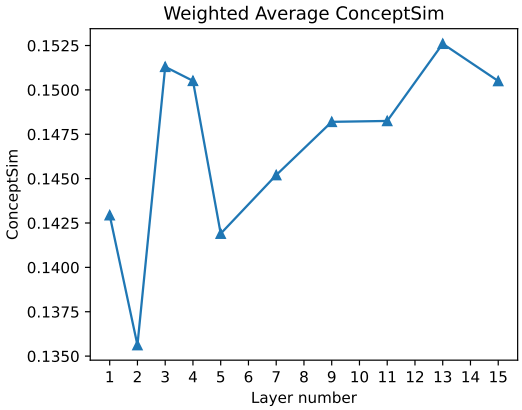


(e) Feature 10 - Category World

Figure 13: Example of a sentence misclassified by our fine-tuned Pythia-1B model on the AG News dataset. The true label is Sport, but the model predicted the World category. The activated concepts computed by HI-Concept are shown, along with their respective attributions over the words in the sentence.

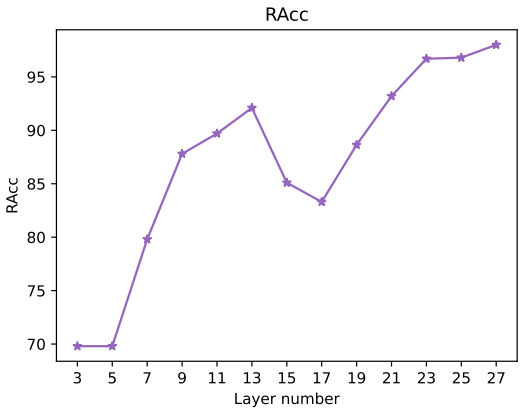


(a) Recovery accuracy (RAcc) of the concepts extracted by `ClassifSAE` across the layers of Pythia-1B fine-tuned on AG News

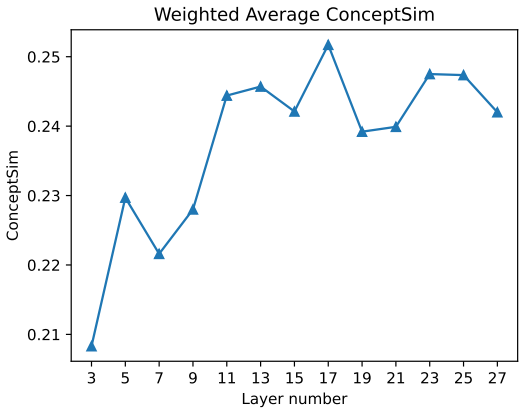


(b) Weighted Average `ConceptSim` of the concepts extracted by `ClassifSAE` across the layers of Pythia-1B fine-tuned on AG News

Figure 14: Comparison of the extracted concepts properties across layers of Pythia-1B fine-tuned on AG News.



(a) Recovery accuracy (RAcc) of the concepts extracted by `ClassifSAE` across the layers of GPT-J fine-tuned on TweetEval Offensive



(b) Weighted Average `ConceptSim` of the concepts extracted by `ClassifSAE` across the layers of GPT-J fine-tuned on TweetEval Offensive

Figure 15: Comparison of the extracted concepts properties across layers of GPT-J fine-tuned on TweetEval Offensive.

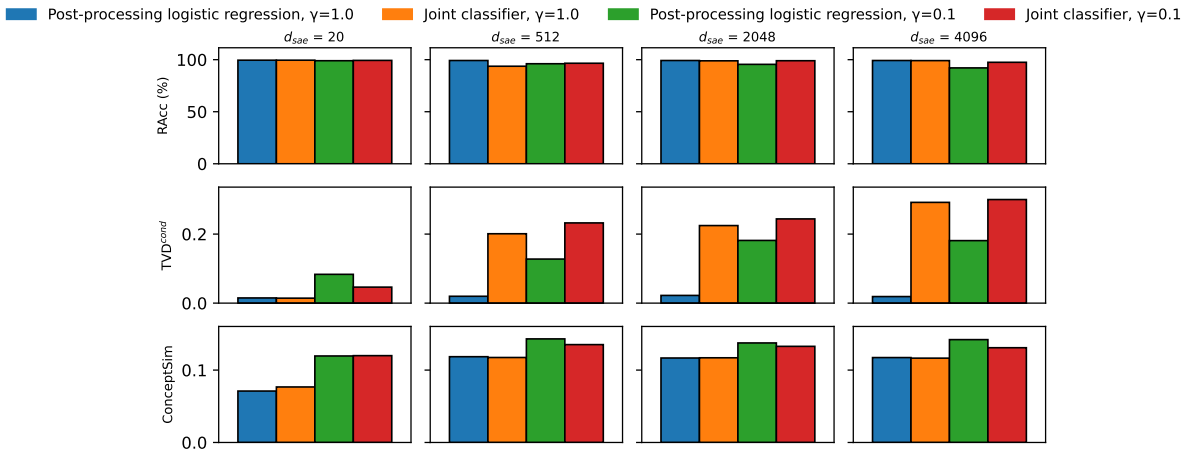


Figure 16: Report of the recovery accuracy RAcc for completeness, TVD$^{\text{cond}}$ for individual causality and `ConceptSim` for interpretability (see Sections 3.2 and 3.3) under ablations of the joint classifier and activation rate sparsity mechanism across varying SAE hidden layer sizes. For d_{sae} fixed, we test four configurations: Regular SAE without either component ($\gamma = 1$, Logistic regression for the selection of $\mathbf{z}_{\text{class}}$), SAE with only the activation rate sparsity mechanism ($\gamma = 0.1$, Logistic regression for the selection of $\mathbf{z}_{\text{class}}$), SAE trained with the joint classifier but no activation rate sparsity loss ($\gamma = 1$, Joint classifier) and `ClassifSAE` ($\gamma = 0.1$, Joint classifier). TVD$^{\text{cond}}$ increases significantly with the inclusion of a learned classifier while `ConceptSim` mostly benefits from the activation rate sparsity loss. We note that while the activation rate sparsity loss enforcement alone maximizes `ConceptSim`, the simultaneous integration of the two components produces the best overall trade-off across the investigated three metrics. Experiments are carried out on residual-stream activations taken at the penultimate layer of Pythia-1B fined-tuned on AG News.

			Classifier Model						
			BERT	DeBERTa-v3	Pythia-410M	Pythia-1B	GPT-J	Mistral-Inst.	Llama-Inst.
Dataset	**Method**	**Metric**							
AG News	ICA	RAcc (%,↑)	99.80	99.98	99.43	99.60	99.68	99.05	96.34
		Δf^{cond} (%,↑)	0.57	1.98	2.47	2.63	2.24	1.76	3.33
		`ConceptSim` (↑)	0.0627	0.0612	0.0609	0.0607	0.0609	0.0662	0.0616
		Avg Act. rate (%,↓)	100	100	100	100	100	100	100
	ConceptSHAP	RAcc (%,↑)	99.58	99.84	97.27	98.55	98.17	96.18	88.68
		Δf^{cond} (%,↑)	11.35	0.76	9.98	13.34	10.06	5.54	1.19
		`ConceptSim` (↑)	0.1226	0.1250	0.1259	0.1260	0.1323	0.1312	0.1430
		Avg Act. rate (%,↓)	26.32	25.71	24.38	24.56	24.06	23.28	20.60
	HI-Concept	RAcc (%,↑)	99.72	99.91	99.63	99.72	99.82	99.67	98.97
		Δf^{cond} (%,↑)	53.03	81.32	7.09	20.47	33.56	31.05	12.81
		`ConceptSim` (↑)	0.1112	0.1239	0.0949	0.0907	0.0823	0.075	0.0852
		Avg Act. rate (%,↓)	33.51	25.93	37.15	47.53	50.97	63.56	58.73
	SAE (K=10)	RAcc (%,↑)	98.63	99.67	98.17	98.89	99.00	96.24	92.63
		Δf^{cond} (%,↑)	0.92	0.27	2.32	2.87	4.54	7.76	10.59
		`ConceptSim` (↑)	0.1297	0.1241	0.1185	0.1264	0.099	0.0696	0.0699
		Avg Act. rate (%,↓)	24.20	26.53	18.54	26.40	30.72	49.93	42.77
	`ClassifSAE` (K=10)	RAcc (%,↑)	99.14	96.30	98.84	99.43	99.39	99.02	96.57
		Δf^{cond} (%,↑)	14.27	23.63	18.67	22.10	24.32	17.86	17.53
		`ConceptSim` (↑)	0.1566	0.1371	0.1562	0.1412	0.1648	0.1431	0.1444
		Avg Act. rate (%,↓)	9.59	9.65	8.41	8.11	8.28	10.28	9.38

Table 5: Completeness, causality and interpretability metrics (see Sections 3.2 and 3.3) of the concepts learned from different LLM classifiers for the dataset AG News. Prior each task evaluation, all models are fine-tuned at the exception of Mistral-Instruct and Llama-Instruct, which are aligned with the task via soft-prompt tuning. Δf^{cond} is simply the average of $\Delta f^{cond}_{\{j\}}$. `ConceptSim` is the average of the individual concept scores, weighted as detailed in the Appendix D. All post-hoc methods were configured to search for 20 concepts. Concepts are computed from the sentence-level hidden state: for decoder-only models, from the residual stream after the penultimate transformer block and for encoder-only models, from the layer preceding the classification head. Results are obtained with seed equals to 42

			Classifier Model						
			BERT	DeBERTa-v3	Pythia-410M	Pythia-1B	GPT-J	Mistral-Inst.	Llama-Inst.
Dataset	**Method**	**Metric**							
TE Offensive	ICA	RAcc (%,↑)	99.77	99.86	99.22	99.08	99.63	98.58	94.05
		Δf^{cond} (%,↑)	1.81	0.84	3.10	4.14	2.72	3.22	1.93
		`ConceptSim` (↑)	0.2059	0.2076	0.2091	0.2044	0.2044	0.2130	0.2127
		Avg Act. rate (%,↓)	100	100	100	100	100	100	100
	ConceptSHAP	RAcc (%,↑)	97.00	97.80	95.55	82.73	95.19	89.88	92.08
		Δf^{cond} (%,↑)	0.06	0.29	0.26	1.03	0.23	0.02	0.03
		`ConceptSim` (↑)	0.2049	0.2015	0.2062	0.2241	0.2128	0.2433	0.2279
		Avg Act. rate (%,↓)	44.78	46.83	44.04	24.95	36.46	10.77	20.12
	HI-Concept	RAcc (%,↑)	99.54	99.77	99.13	99.63	98.99	99.16	98.90
		Δf^{cond} (%,↑)	1.66	11.28	8.47	12.20	2.67	8.45	1.79
		`ConceptSim` (↑)	0.2073	0.2027	0.2048	0.2051	0.2049	0.2062	0.2045
		Avg Act. rate (%,↓)	38.92	53.10	53.00	36.24	57.98	64.37	64.24
	SAE (K=10)	RAcc (%,↑)	98.07	99.08	97.53	97.57	85.89	88.28	94.30
		Δf^{cond} (%,↑)	0.72	0.49	1.98	8.41	17.45	6.65	18.93
		`ConceptSim` (↑)	0.2199	0.2079	0.2071	0.2024	0.2034	0.2097	0.2073
		Avg Act. rate (%,↓)	13.22	34.79	12.52	10.14	43.05	34.40	35.96
	`ClassifSAE` (K=10)	RAcc (%,↑)	97.76	99.35	97.66	94.10	93.17	96.75	96.79
		Δf^{cond} (%,↑)	10.34	12.40	9.70	4.69	9.60	7.34	10.75
		`ConceptSim` (↑)	0.2488	0.2342	0.2264	0.2328	0.2379	0.2489	0.2224
		Avg Act. rate (%,↓)	9.13	9.04	6.77	9.88	10.19	6.85	8.77

Table 6: Completeness, causality and interpretability metrics (see Sections 3.2 and 3.3) of the concepts learned from different LLM classifiers for the dataset TweetEval Offensive. Prior each task evaluation, all models are fine-tuned at the exception of Mistral-Instruct and Llama-Instruct, which are aligned with the task via soft-prompt tuning. Δf^{cond} is simply the averages of $\Delta f^{cond}_{\{j\}}$. `ConceptSim` is the average of the individual concept scores, weighted as detailed in the Appendix D. All post-hoc methods were configured to search for 20 concepts. Concepts are computed from the sentence-level hidden state: for decoder-only models, from the residual stream after the penultimate transformer block and for encoder-only models, from the layer preceding the classification head. Results are obtained with seed equals to 42.

			Classifier Model						
			BERT	DeBERTa-v3	Pythia-410M	Pythia-1B	GPT-J	Mistral-Inst.	Llama-Inst.
Dataset	**Method**	**Metric**							
	ICA	RAcc (%,↑)	98.58	99.37	98.51	97.99	98.73	92.12	84.40
		Δf^{cond} (%,↑)	4.78	4.31	4.03	5.36	5.08	5.46	7.22
		`ConceptSim` (↑)	0.1146	0.1154	0.1182	0.1170	0.1161	0.1216	0.1219
		Avg Act. rate (%,↓)	100	100	100	100	100	100	100
	ConceptSHAP	RAcc (%,↑)	89.70	95.10	91.65	95.20	91.47	94.62	81.39
		Δf^{cond} (%,↑)	11.81	1.10	5.81	1.21	1.30	7.91	7.14
		`ConceptSim` (↑)	0.1453	0.1297	0.1171	0.1332	0.1324	0.1516	0.1565
		Avg Act. rate (%,↓)	33.57	31.39	30.85	28.45	25.19	28.01	17.11
TE Sentiment	HI-Concept	RAcc (%,↑)	98.32	99.00	98.91	98.63	99.03	98.41	95.62
		Δf^{cond} (%,↑)	25.43	18.69	12.56	9.02	9.23	8.80	14.12
		`ConceptSim` (↑)	0.1225	0.1237	0.1246	0.12137	0.1206	0.1217	0.1259
		Avg Act. rate (%,↓)	39.53	38.04	46.79	46.31	53.62	62.34	50.48
	SAE (K=10)	RAcc (%,↑)	93.85	95.45	87.05	96.26	94.11	88.80	70.56
		Δf^{cond} (%,↑)	6.05	3.04	3.38	5.42	7.80	16.48	28.84
		`ConceptSim` (↑)	0.1465	0.1344	0.1518	0.1293	0.1229	0.1174	0.1263
		Avg Act. rate (%,↓)	14.03	27.62	15.49	15.38	39.18	44.56	45.13
	`ClassifSAE` (K=10)	RAcc (%,↑)	94.20	97.89	95.90	96.21	95.91	91.44	93.23
		Δf^{cond} (%,↑)	10.09	15.90	13.04	15.06	12.16	14.26	19.72
		`ConceptSim` (↑)	0.1577	0.1494	0.1458	0.1646	0.1574	0.1550	0.1500
		Avg Act. rate (%,↓)	9.62	8.89	8.96	8.30	8.69	8.95	15.27

Table 7: Completeness, causality and interpretability metrics (see Sections 3.2 and 3.3) of the concepts learned from different LLM classifiers for the dataset TweetEval Sentiment. Prior each task evaluation, all models are fine-tuned at the exception of Mistral-Instruct and Llama-Instruct, which are aligned with the task via soft-prompt tuning. Δf^{cond} is simply the averages of $\Delta f^{cond}_{\{j\}}$. `ConceptSim` is the average of the individual concept scores, weighted as detailed in the Appendix D. All post-hoc methods were configured to search for 20 concepts. Concepts are computed from the sentence-level hidden state: for decoder-only models, from the residual stream after the penultimate transformer block and for encoder-only models, from the layer preceding the classification head. Results are obtained with seed equals to 42.

			Classifier Model						
			BERT	DeBERTa-v3	Pythia-410M	Pythia-1B	GPT-J	Mistral-Inst.	Llama-Inst.
Dataset	**Method**	**Metric**							
IMDB	ICA	RAcc (%,↑)	99.94	99.98	99.98	99.92	99.96	97.21	99.50
		Δf^{cond} (%,↑)	2.78	2.57	2.06	2.37	2.29	3.72	2.51
		`ConceptSim` (↑)	0.2988	0.2956	0.2963	0.3012	0.2997	0.3005	0.3005
		Avg Act. rate (%,↓)	100	100	100	100	100	100	100
	ConceptSHAP	RAcc (%,↑)	99.74	99.40	99.48	99.75	99.74	93.34	94.52
		Δf^{cond} (%,↑)	3.30	0.00	0.00	0.21	0.00	0.43	1.16
		`ConceptSim` (↑)	0.3012	0.3031	0.3030	0.3051	0.3046	0.3083	0.3109
		Avg Act. rate (%,↓)	45.70	42.85	49.39	48.71	44.00	28.02	42.67
	HI-Concept	RAcc (%,↑)	99.77	99.97	99.94	99.90	99.94	98.69	99.58
		Δf^{cond} (%,↑)	38.53	24.05	39.14	25.45	59.33	17.143	0.48
		`ConceptSim` (↑)	0.3068	0.3049	0.3082	0.3021	0.2925	0.3013	0.3017
		Avg Act. rate (%,↓)	44.69	47.71	50.15	47.46	73.88	57.05	71.85
	SAE (K=10)	RAcc (%,↑)	99.01	99.68	99.42	99.60	98.85	91.79	98.12
		Δf^{cond} (%,↑)	0.47	9.94	0.24	0.44	6.12	15.54	6.18
		`ConceptSim` (↑)	0.3096	0.3139	0.3099	0.3043	0.3101	0.3042	0.3003
		Avg Act. rate (%,↓)	30.12	27.84	27.06	31.24	29.02	49.52	43.76
	`ClassifSAE` (K=10)	RAcc (%,↑)	98.88	99.87	95.57	94.02	99.78	96.51	98.84
		Δf^{cond} (%,↑)	4.76	3.92	16.53	17.85	2.05	5.76	22.26
		`ConceptSim` (↑)	0.3045	0.3036	0.3031	0.3052	0.3104	0.3087	0.3130
		Avg Act. rate (%,↓)	9.76	9.96	8.27	7.66	11.50	13.34	7.13

Table 8: Completeness, causality and interpretability metrics (see Sections 3.2 and 3.3) of the concepts learned from different LLM classifiers for the dataset IMDB. Prior each task evaluation, all models are fine-tuned at the exception of Mistral-Instruct and Llama-Instruct, which are aligned with the task via soft-prompt tuning. Δf^{cond} is simply the averages of $\Delta f^{cond}_{\{j\}}$. `ConceptSim` is the average of the individual concept scores, weighted as detailed in the Appendix D. All post-hoc methods were configured to search for 20 concepts. Concepts are computed from the sentence-level hidden state: for decoder-only models, from the residual stream after the penultimate transformer block and for encoder-only models, from the layer preceding the classification head. Results are obtained with seed equals to 42