

A Financial Brain Scan of the LLM

Hui Chen,^{*} Antoine Didisheim,[†] Mohammad (Mo) Pourmohammadi,[‡]
Luciano Somoza,[§] Hanqing Tian[¶]

March, 2026

Abstract

Emerging techniques in computer science make it possible to “brain scan” large language models (LLMs), identify the plain-English concepts that guide their reasoning, and steer them while holding other factors constant. We show that this approach can map LLM-generated economic forecasts to concepts such as sentiment, technical analysis, and timing, and compute their relative importance without reducing performance. We also show that models can be steered to be more or less risk-averse, optimistic, or pessimistic, which allows researchers to correct or simulate biases. The method is transparent, lightweight, and replicable for empirical research in the social sciences.

^{*}MIT Sloan and NBER. Email: huichen@mit.edu

[†]University of Melbourne. Email: antoine.didisheim@unimelb.edu.au

[‡]Yale SOM. Email: mo.pourmohammadi@yale.edu

[§]ESSEC Business School. Email: somoza@essec.edu

[¶]University of Melbourne. Email: hanqing.tian1@unimelb.edu.au

I. Introduction

Research on artificial intelligence is growing rapidly. Large language models (LLMs) are already part of financial analysis, research workflows, and trading strategies (Cheng, Lin, and Zhao, 2024). Their appeal is clear: they process text at scale, summarize efficiently, and produce consistent answers from noisy inputs. However, there are two main concerns for researchers in economics and finance. First, LLMs’ scale and density make them uninterpretable black boxes, limiting their usefulness for research (Ludwig, Mullainathan, and Rambachan, 2025). Second, LLMs contain hard-to-identify biases, for instance tilting output toward specific demographic preferences (Fedyk, Kakhbod, Li, and Malmendier, 2024). Thus, it is essential to develop and apply methods that increase transparency and align model behavior with the researchers’ objectives.

This paper applies an emerging technique in computer science (see, e.g., Cunningham et al., 2023; Gao, 2024; Shi, 2025) to economic tasks. The technique allows opening and controlling an LLM by inserting an interpretable sparse representation within its architecture, meaning a flat layer (i.e., vector) with a direct mapping between the nodes and words in plain English.

Through this representation, a researcher can identify the concepts the model uses to process a given input, expressed in plain English. The array of concepts is extremely large and ranges from optimism or financial risk to fondness for specific cuisines. Furthermore, by adjusting the code, we can manually regulate the intensity at which the model “thinks” about any targeted concept while keeping everything else equal. The capability of observing an LLM’s “mental” process and manually steering it towards any direction are absent in current financial research and can be valuable for applications across all social sciences.

To showcase the technique, we ask the LLM to allocate 100 dollars between bonds and the S&P500 multiple times, each time adjusting the steering coefficient of the model’s “financial risk” feature (see Figure 1). The more we manually force the model to activate the “financial risk” feature while processing an answer, the more it allocates resources towards bonds, consistent with an increase in risk aversion.

In the evolution of the human brain, incentives have shaped specialized pathways, enabling the study of brain activity through scans that reveal which regions are engaged by specific tasks. In contrast, LLMs are not trained under such biologically grounded incentives, making their inner representation dense and therefore uninterpretable. The technique

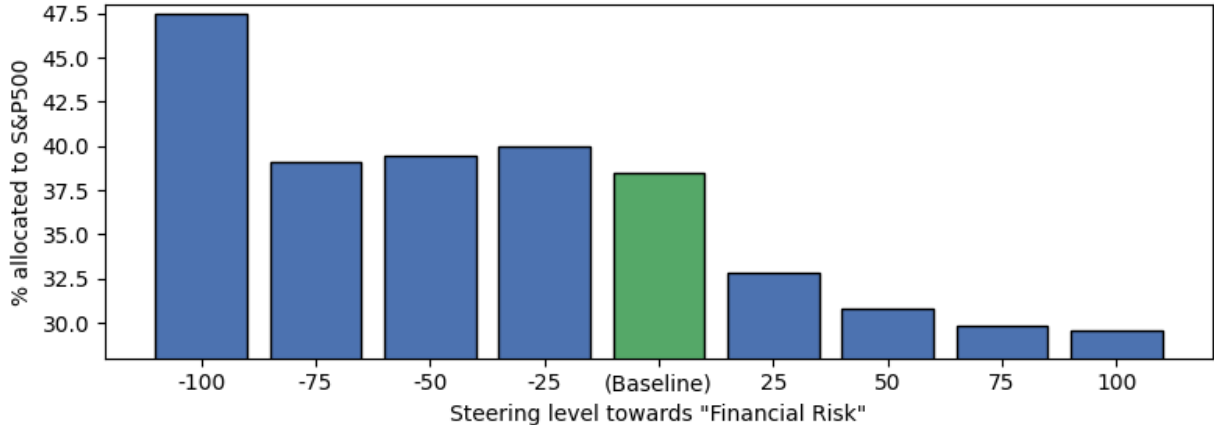


Figure 1. Illustration: Steering LLM’s Risk Aversion

We prompt the LLM to allocate \$100 between the S&P500 and bonds. We then vary the intensity with which the model is steered to activate the “financial risk” feature (x-axis) and record the resulting share allocated to the S&P500 (y-axis). To reduce variance, the experiment is repeated across 100 random seeds, and we report the averaged outcomes.

used in this paper solves this problem by introducing a sparse embedding¹ within an already trained² LLM. All information flowing through the model is encoded into this representation and subsequently decoded. This encoder-decoder structure, referred to as a Sparse Auto-Encoder (SAE), is trained to be benign and not impact the LLMs’ quality. In a second step, generalist corpus of texts are passed throughout the SAE-augmented LLM to label each feature on the sparse representation. In lay terms, just like neuroscientists record which brain regions “light up” when a person sees a face or hears music, here we look at which features in the model activate when it processes certain words or contexts. In both cases, the system is not labeled in advance but the researcher discovers meaning by observing consistent patterns of activation.

We divide our analysis in two equally important parts. First, we focus on interpretability and use the sparse embedding as input to a forecasting model, to understand what generates the LLM-driven excess returns documented in recent papers (Chen, Kelly, and Xiu, 2022;

¹LLMs process information through an iterative hierarchy of layers, where each layer produces a vector representation (an embedding) that encodes the model’s internal representation of the input text. A sparse embedding is characterized by having non-zero activations on only a limited subset of its dimensions. This sparsity is crucial for interpretability, as it constrains the model to associate specific and localized dimensions with distinct semantic features. In contrast, dense embeddings represent concepts as distributed patterns across all dimensions, making their internal structure more entangled and less amenable to direct interpretation.

²Training an LLM means estimating the values of its parameters, which is an extremely costly endeavor. Retraining the model to incorporate a sparse representation would therefore be computationally prohibitive. Fortunately, a Sparse Autoencoder (SAE) can be integrated with an already trained LLM, enabling the extraction of sparse representations without the need for full retraining.

Lopez-Lira and Tang, 2023a; Chen, Didisheim, and Somoza, 2024). Second, we show how steering can detect and correct biases in canonical finance applications and how this technique can be applied in broader settings to tune LLM’s preferences and biases.

A. *Interpretable LLMs’ embeddings*

In their seminal review, Gentzkow, Kelly, and Taddy (2019) observe that a large fraction of empirical applications of textual analysis in economics and finance can be summarized in three broad steps: (i) constructing a numerical representation of the text; (ii) mapping these representations to predicted values of an unknown outcome; and (iii) employing the predicted values in subsequent descriptive or causal analyses.

The first step (numerical representation) entails a trade-off between interpretability and efficiency. Simpler methods, such as dictionary-based approaches and bag-of-words models, provide highly transparent and interpretable representations (Tetlock, 2007; Loughran and McDonald, 2011). On the one hand, such interpretability is crucial for academics seeking to understand underlying mechanisms, as well as for practitioners, for whom opaque models may entail mechanical exposure to rare, high-impact “black swan” events. On the other hand, Chen et al. (2022) demonstrate that complex, opaque embeddings generated by modern machine learning methods (see, e.g., Sarkar, 2025) can significantly outperform simpler representations in important financial prediction tasks.

The methodology proposed in this paper provides both interpretability and performance. Using sparse auto-encoders, any prompt passed through an LLM can be mapped to a semantically rich sparse representation, which is a vector that captures the internal representation of prompts with two key properties: *sparsity*, meaning that only a small subset of its dimensions are nonzero at any given time, and *semantic loading*, meaning that each dimension corresponds to a distinct and interpretable semantic concept, such as “risks” or “optimism.” This sparse vector can then be applied in step (i) of the Gentzkow et al. (2019) framework.

To assess the economic significance of these representations, we adapt the framework of Chen et al. (2022) and extract the sparse representation of each news item in a sample of Reuters articles from 2015–2024. Because the resulting vectors are high-dimensional, we first apply principal component logistic regression (Aguilera, Escabias, and Valderrama, 2006) — a standard statistical dimension-reduction technique to retain the 5,000 most relevant features.³

We first show that sparse representations capture economic information as well as, or bet-

³Specifically, we train a logistic model on the first 1,000 principal components, then back-project the model coefficients to the original feature space to rank variables by importance.

ter than, state-of-the-art models. [Chen et al. \(2022\)](#) provides a natural benchmark. They use the last layer of LLMs as embeddings⁴ and show that this representation outperforms all previous textual methods in the finance literature. To compare our reduced sparse embeddings with the last-layer embeddings from the same LLM, we follow [Chen et al. \(2022\)](#) and train two models, one with each embedding, to predict the next day’s return.⁵ Comparing the Sharpe ratios of an intraday long–short strategy, the benchmark model attains a Sharpe ratio of 4.91, whereas the sparse representation achieves 5.51.

We then investigate whether a subset of features captures most of the predictability. We rank features in the sparse vector by importance, based on the absolute loadings of the logistic model from the previous exercise. Features with large absolute loadings strongly affect predictions, while those near zero have little impact. We repeat the prediction exercise—training logistic regressions on a rolling window and building a long-short strategy from the forecasts—using subsets of the most important features. Consistent with the notion of “Virtue of Complexity” [Kelly, Malamud, and Zhou \(2024\)](#), adding features always increases the Sharpe ratio. Still, a model trained on only the 5 most important features achieves a Sharpe ratio of 3.34, while one trained on 300 out of 5,000 features reaches 5.21.

Having shown that efficiency is not compromised and that all features contribute to maximum performance, we now turn to the main advantage of sparse representations: interpretability. Each feature in a sparse representation corresponds to a specific semantic meaning. In this paper, we use the pre-trained and labeled sparse model released by Google DeepMind ([Lieberum et al., 2024](#)).⁶ The features’ labels are not finance-specific but reflect general applications. Some labels capture subtle distinctions between related concepts. For example, one feature is labeled “negative sentiments conditions acceptance limitations,” while another is labeled “expressing opinions judgments performance events.” Although these can be viewed as distinct categories, it may be more appropriate to group them under broader “sentiment” categories. Thus, constructing groups of labels is a natural step.

To solve this problem we propose a new methodology to group the features into economically relevant clusters. In line with [Bybee, Kelly, Manela, and Xiu \(2024\)](#), we employ unsupervised learning to construct these groups. Specifically, we apply k -means clustering to

⁴Large language models consist of many layers that gradually transform text into numerical representations capturing deeper meaning. The final (last) layer produces the most refined version of this representation, summarizing the semantic content of the text into an “embedding” that can be used for prediction tasks as in [Chen et al. \(2022\)](#).

⁵For both models, we use logistic regression, trained on a three-year rolling window with an additional one-year validation set and re-estimated annually.

⁶The features in this model are labeled using the methodology described in Section II.

rich embeddings of the feature labels.⁷ The methodology produces 17 clusters with distinct economic interpretation.

This separation enables the construction of two model types and their corresponding long-short portfolios: (i) models trained exclusively on features from a single group, and (ii) models trained on all but one group, following the Shapley value framework (Gu, Kelly, and Xiu, 2020). Comparing these models with the *full feature* model, which jointly incorporates all 17 groups, allows us to quantify the contribution of individual financial concepts to LLM performance in portfolio applications.

Unsurprisingly, the most important feature group is *Sentiment*, closely followed by *Market/Finance* and *Technical Analysis* concepts. Interestingly, time-related features (e.g., dates, years, timelines) exhibit the fourth-highest Shapley value but the lowest individual Sharpe ratios, i.e., the Sharpe ratio obtained when training solely on temporal notions. This observation is consistent with the idea that LLMs capture timing information and leverage it to distinguish between news with short- and long-term impact. Hence, temporal features are crucial for achieving maximal performance, yet remain uninformative in isolation, as they do not provide any directional signal.

B. Steering and Bias Correction

Having shown how sparse autoencoders increase the transparency of LLMs, we next highlight the second main advantage: they allow us to force the model to incorporate a specific concept, with a chosen intensity, when processing an input.

We start by building a long-short trading strategy based on prompt-extracted sentiment from news in the spirit of Lopez-Lira and Tang (2023a) and Chen et al. (2024). We employ an LLM to estimate the sentiment of aftermarket and overnight Reuters news concerning individual stocks. On the sparse representation, we select a feature linked to “positivity”. For each news item, we obtain a baseline LLM forecast and additional forecasts corresponding to varying levels of positive steering. As expected, the proportion of positive classifications increases monotonically with stronger positive steering and the conditional returns are consistent with the model’s induced interpretation of the same headline. In the spirit of Lopez-Lira and Tang (2023a), we use these forecasts to construct intra-day long-short portfolios, going long (equally weighted) on stocks with positive sentiment and short on

⁷Bybee et al. (2024) employ a large corpus of news texts rich in linguistic nuance, making the LDA algorithm suitable. By contrast, our task involves clustering a relatively small set of approximately 5,000 feature labels, which is insufficient to learn robust textual structures. Hence, following the approach of Sarkar (2025), we rely on a pre-trained embedding model, which is more appropriate for our setting.

those with negative sentiment. Interestingly, negatively steered predictions achieve a statistically significant higher annualized Sharpe ratio (4.28) compared to the baseline (3.87).⁸ Notably, all negatively steered predictions yield higher Sharpe ratios than both the baseline and positively steered predictions, with the effect most pronounced for moderate steering.

These results suggest two points. First, LLM forecasts are biased toward positive sentiment. While this finding may not generalize to all models, it is consistent with evidence in the literature (Fatahi, Vassileva, and Roy, 2024). Second, steering can be used to correct this bias.

In principle, this bias-correction framework can be applied to any LLM bias or preference that can be mapped onto the set of features in the sparse representation, either to attenuate or to amplify such tendencies. We illustrate this by steering features labeled as risk aversion and attention to wealth, and subsequently prompting the model to choose between safe and risky investments or to allocate a budget between Treasuries and an equity index. Repeated prompting yields monotonic shifts in behavior consistent with the intended steering direction: stronger risk aversion decreases equity allocations and increases the share of safe choices, whereas stronger attention to wealth produces the opposite effect. This provides a practical mechanism to simulate agents with configurable preferences without retraining the base model.

Our contribution is twofold. First, we propose a method to group sparse autoencoder features into economically meaningful clusters. We show that this method can make LLMs transparent for economic analysis, allowing researchers to identify the concepts driving predictions without reducing performance. Second, we find that these interpretable features enable precise control over model behavior (such as sentiment or risk aversion), providing a practical approach to correct biases, simulate preferences, and design controlled experiments. Together, these findings offer a lightweight, replicable method for turning LLMs from opaque systems into useful tools for empirical research in economics and the social sciences.

Related Literature Modern asset-pricing research shows that flexible, high-dimensional representations can improve prediction and interpretation. Gu et al. (2020) find that machine learning methods extract return-relevant structure from large predictor sets. For this paper’s interpretability goal, representation learning with autoencoders can summarize risk exposures while remaining economically transparent. This approach applies that idea by replacing dense, opaque LLM embeddings with a sparse, interpretable feature set, then using those features to steer the model’s output.

⁸We follow the approach proposed by Jobson and Korkie (1981), incorporating the correction noted in Memmel (2003), to test the significance of our results. In addition, we show that the negatively steered portfolio generates statistically significant alpha relative to the baseline strategy.

The most related study is [Chen et al. \(2022\)](#), who use contextualized news embeddings from large language models to forecast next-day stock returns in U.S. and international markets, reporting sizable gains over bag-of-words and other NLP baselines. Building on that work, this paper retains predictive power but replaces dense embeddings with a sparse, interpretable feature set, showing that most of the signal loads on refined sentiment. The design also complements prompt-based methods that classify news directly ([Lopez-Lira and Tang, 2023b](#)), and it follows best-practice evaluation to avoid training-cutoff and look-ahead errors ([Sarkar and Vafa, 2024](#)). It also connects to [Bybee, Kelly, Manela, and Xiu \(2020\)](#) by documenting systematic variation in the optimal correction across topics and industries.

Furthermore, our paper speaks to the fast-growing line of work that treats LLMs as stand-ins for human subjects. [Horton \(2023\)](#) argues that LLMs can serve as “*homo silicus*” in economic simulations. [Aher, Arriaga, and Kalai \(2023\)](#) formalize “Turing Experiments” and show that recent models replicate classic findings in behavioral economics and psychology. [Argyle, Busby, Fulda, Gubler, Rytting, and Wingate \(2023\)](#) demonstrate that conditioning LLMs on demographics can reproduce group-level survey response distributions. In HCI, [Park, O’Brien, Cai, Morris, Liang, and Bernstein \(2023\)](#) build multi-agent, memory-based “generative agents” that exhibit believable individual and social behavior. Our contribution to this agenda is methodological: we show how to produce heterogeneous LLMs that vary along a *single* interpretable dimension-e.g., positivity or risk aversion-while holding other latent factors fixed, enabling clean comparative statics in finance tasks and portable designs for the social sciences.

II. Methodology

The technology underlying generative models like chat-GPT and similar LLMs, is based on *generative architectures* that model the probability distribution of text sequences. At their core, these models estimate the conditional probability of the next token, typically a word or subword, given the preceding sequence of text.

These generative models rely on stacked *transformer blocks*. Each block applies a complex transformation to the output of the previous block (the model’s internal state) and passes the updated state to the next layer. This layered design lets the model gradually learn more complex language patterns as information moves up through the layers. (see, e.g. [Vaswani et al., 2017](#); [Radford et al., 2019](#)).

Formally,⁹ let us denote the internal state of the model at layer l as $\mathbf{r}^{(l)}$, a vector in $\mathbb{R}^d$ for some d . This vector, often called the *residual stream*, serves as a running summary of the information accumulated up to that point in the sequence. Each transformer block, denoted by $\text{Block}^{(l)}$, receives the current residual stream, applies a learned transformation, and provides a modification to the stream. The architecture is structured such that this modification is added back to the original input:

$$\mathbf{r}^{(l+1)} = \mathbf{r}^{(l)} + \text{Block}^{(l)}(\mathbf{r}^{(l)}). \quad (1)$$

This recursive formulation captures the essential mechanism, where each layer refines the current representation by processing and reintegrating new information. With m total blocks, the final residual stream $\mathbf{r}^{(m)}$ is passed to an output module that maps it to a probability distribution over the vocabulary. This distribution represents the model’s prediction for the next word:

$$P(\text{Next Word} \mid \text{Input Text}) = \text{Block}^{(\text{Output})}(\mathbf{r}^{(m)}). \quad (2)$$

This architecture, shared by many state-of-the-art models, has demonstrated remarkable performance across a wide range of NLP benchmarks (e.g., [OpenAI, 2024](#); [Llama, 2024](#); [Gemini, 2025](#)), yet it remains notoriously difficult to interpret. The source of this opacity is twofold. First, the individual transformation blocks are large, nonlinear, and densely parameterized, which makes them uninterpretable to a human. Second, the residual streams are dense. Individual concepts like “optimism” or “pessimism” are unlikely to be captured by a single point on the stream. Instead, individual concepts are captured by a complex combination of many individual points. In the human brain, different regions often specialize for distinct tasks such as language, vision, or motor control, making it easier to associate structure with function. Large language models, by contrast, have no such architectural or evolutionary incentive to develop specialized pathways. Their sole training objective is to predict the next token, which can be met using highly distributed representations with no clear division of labor across dimensions.

This lack of interpretability has been a persistent concern, especially in high-stakes or domain-sensitive applications ([Ludwig et al., 2025](#)). Without insight into what information is stored in $\mathbf{r}^{(l)}$ or how it evolves across layers, understanding how LLMs process financial information or tweaking this process is virtually impossible.

⁹For simplicity, we omit the input length dimension and present the model as if processing a single token at a time. This simplification does not affect the core intuition and is used purely for notational convenience.

To address this, we seek to discover a sparse and semantically meaningful representation of the residual stream. Specifically, we aim to identify a transformation:

$$\mathbf{r}^{(l)} \mapsto \mathbf{z}^{(l)} \in \mathbb{R}^k$$

where each coordinate of $\mathbf{z}^{(l)}$ corresponds to an *interpretable* feature or concept. Intuitively, this would enable us to “open the black box” by exposing a layer-wise semantic decomposition of the model’s internal reasoning.

A. Sparse Autoencoders (SAEs)

Autoencoders are a class of neural network models designed for unsupervised dimensionality reduction. Like Principal Component Analysis (PCA), they aim to compress input data into a lower-dimensional latent representation which minimizes the reconstruction error. However, whereas PCA minimizes reconstruction error using linear projections onto orthogonal principal components, autoencoders directly minimize the reconstruction error without imposing linearity or orthogonality constraints, allowing them to learn more flexible and potentially more powerful nonlinear mappings. Their ability to learn nonlinear embeddings has made them a powerful tool for data compression, feature extraction, and anomaly detection. In finance, autoencoders have been applied to problems such as factor discovery and risk modeling (see, e.g., [Gu, Kelly, and Xiu, 2021](#)).

Formally, an autoencoder consists of two components: an encoder and a decoder. Given an input vector $\mathbf{x} \in \mathbb{R}^d$, the encoder maps it to a latent representation $\mathbf{z} \in \mathbb{R}^k$ via a learned transformation:

$$\mathbf{z} = f(\mathbf{W}_e \mathbf{x} + \mathbf{b}_e), \quad (3)$$

where $\mathbf{W}_e \in \mathbb{R}^{k \times d}$ is the encoder weight matrix, $\mathbf{b}_e \in \mathbb{R}^k$ is a bias vector, and $f(\cdot)$ is a nonlinear activation function.¹⁰ For simplicity, we present here the case of a single-layer encoding and decoding model. However, as with any neural network, these layers can be stacked on top of one another.

The decoder then attempts to reconstruct the original input from $\mathbf{z}$:

$$\hat{\mathbf{x}} = g(\mathbf{W}_d \mathbf{z} + \mathbf{b}_d), \quad (4)$$

where $\mathbf{W}_d \in \mathbb{R}^{d \times k}$ and $\mathbf{b}_d \in \mathbb{R}^d$ are decoder parameters, and $g(\cdot)$ is usually chosen to be

¹⁰While k is typically smaller than d in standard autoencoders, in the context of LLM interpretability it is common to set $k > d$ to allow for overcomplete, disentangled representations and lower reconstruction error.

the identity function or a sigmoid, depending on the domain of the input data (Goodfellow, Bengio, Courville, and Bengio, 2016).

The encoder and decoder are trained jointly to minimize the reconstruction error between the input $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$:

$$\mathcal{L}_{\text{AE}}(\mathbf{x}) = \underbrace{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}_{\text{Reconstruction loss}}. \quad (5)$$

Autoencoders are powerful because they can compress data into a lower-dimensional representation, addressing one of the two sources of opacity we discussed earlier: the sheer size and complexity of the transformation blocks. However, they do not address the other major challenge: in order to be interpretable this new smaller representation has to be sparse. This is where *Sparse Autoencoders (SAEs)* can play a role. A SAE is a variant of the standard autoencoder that adds a sparsity-inducing penalty, typically an ℓ_1 norm on the latent representation $\mathbf{z}$. The aim is to encourage most dimensions of $\mathbf{z}$ to be inactive (conceptually zero for any given input), thereby promoting interpretability and disentanglement in the learned features:

$$\mathcal{L}_{\text{SAE}}(\mathbf{x}) = \underbrace{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}_{\text{Reconstruction}} + \underbrace{\lambda \|\mathbf{z}\|_1}_{\text{Sparsity penalty}}, \quad (6)$$

where $\lambda > 0$ controls the strength of the sparsity constraint.

This sparse structure is particularly attractive for interpretability: if only a small number of latent features are active for a given residual stream $\mathbf{x}$, then those features can be more easily inspected and understood in isolation.

B. Training Sparse Autoencoders within a Large Language Model

The concept of training Sparse Autoencoders (SAEs) within a Large Language Model (LLM) to enhance interpretability was introduced by Cunningham et al. (2023).

In this and the following section, we describe how SAEs can be used to uncover semantically meaningful structures in the internal representations of LLMs. As illustrated in Figure 2, the objective is to construct an augmented model pipeline in which, at each layer l , the residual stream $\mathbf{r}^{(l)}$ is mapped to a sparse code $\mathbf{z}^{(l)}$. The individual dimensions of $\mathbf{z}^{(l)}$ are designed to correspond to interpretable concepts or functions.

To achieve this, SAEs are trained on the residual streams produced at a fixed layer l in the language model. More precisely an input text is run through the LLM up to layer l . The corresponding residual stream vector (i.e., the current internal representation of the input

at a given layer in the model’s processing) $\mathbf{r}^{(l)} \in \mathbb{R}^d$ is then recorded. Repeating this process across a large corpus gives a dataset of residual streams:

$$\mathcal{D} = \left\{ \mathbf{r}_i^{(l)} \right\}_{i=1}^N,$$

where each $\mathbf{r}_i^{(l)}$ is the residual stream at layer l for the i -th text input.

A sparse autoencoder is then trained to compress and reconstruct each $\mathbf{r}_i^{(l)}$ through a sparse latent representation:

$$\mathbf{z}_i^{(l)} = f(\mathbf{W}_e \mathbf{r}_i^{(l)} + \mathbf{b}_e), \quad (7)$$

$$\hat{\mathbf{r}}_i^{(l)} = g(\mathbf{W}_d \mathbf{z}_i^{(l)} + \mathbf{b}_d), \quad (8)$$

where $f(\cdot)$ is a nonlinear activation function, and $g(\cdot)$ is usually the identity.

The loss function encourages both accurate reconstruction and sparsity in the hidden representation:

$$\mathcal{L}(\mathbf{r}_i^{(l)}) = \underbrace{\|\mathbf{r}_i^{(l)} - \hat{\mathbf{r}}_i^{(l)}\|_2^2}_{\text{Reconstruction loss}} + \lambda \underbrace{\|\mathbf{z}_i^{(l)}\|_1}_{\text{Sparsity penalty}}, \quad (9)$$

with $\lambda > 0$ controlling the trade-off between fidelity and interpretability.

Since the autoencoder is a neural network without a closed-form solution, it is trained by minimizing the expected loss across the generated dataset of residual streams.¹¹

$$\min_{\Theta} \mathbb{E}_{\mathbf{r} \sim \mathcal{D}}[\mathcal{L}(\mathbf{r})], \quad (10)$$

C. Assigning Meaning to Features

With a well-trained SAE we can obtain a sparse representation $\mathbf{z}^{(l)} \in \mathbb{R}^k$ for each residual stream at layer l . Each coordinate $z_j^{(l)}$ in this vector reflects the activation of a learned feature or direction in the model’s latent space. However, while the model learns to use these features during training, they do not come with predefined semantic labels. This section explains how to load each feature of the sparse representation $z_j^{(l)}$ with semantic meaning.

The sparse representation is made of individual feature which can be activated (non-zero) or non activated (zero) when processing any given text. Formally, for a given input residual

¹¹These residual streams are obtained from forward passes of the pretrained language model over text corpora, such as books, news articles, and web pages, similar to those used in LLM training but typically drawn from a smaller, curated subset.

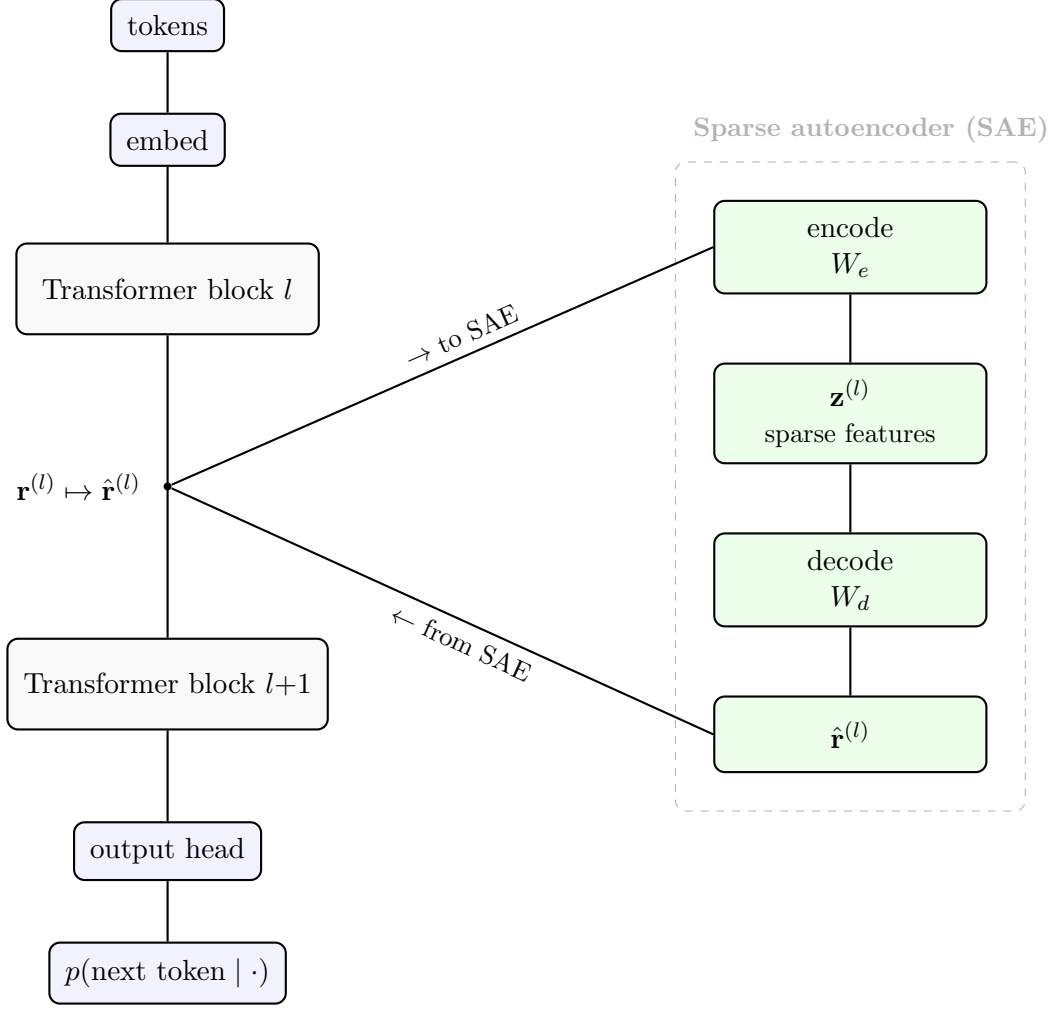


Figure 2. Illustration of SAE’s integration with a Transformer model.

The residual stream $\mathbf{r}^{(l)}$ is extracted from an intermediate layer of the language model and projected onto a sparse, interpretable representation $\mathbf{z}^{(l)}$. This sparse code is then decoded to reconstruct $\hat{\mathbf{r}}^{(l)}$, which is fed back into the model for subsequent processing.

stream $\mathbf{r}^{(l)}$, the activation of feature j is computed as:

$$z_j^{(l)} = f \left([\mathbf{W}_e \mathbf{r}^{(l)} + \mathbf{b}_e]_j \right), \quad (11)$$

where $[\cdot]_j$ extracts the j -th component, and f is the activation function used in the encoder.

For each feature $z_j^{(l)}$, we identify the top- n residual streams in the dataset that lead to the highest absolute values. We then trace these residual streams back to the original text inputs that produced them. This allows us to examine what kinds of tokens, phrases, or contexts consistently trigger high activations. In lay terms, just like neuroscientists record which brain regions “light up” when a person sees a face or hears music, here we look at which features in the model activate when it processes certain words or contexts. In both

cases, the system is not labeled in advance but the researcher discovers meaning by observing consistent patterns of activation.

Example. Suppose a feature $z_j^{(l)}$ frequently shows high activation when the input contains references to years, such as “1998,” “2021,” or “18th century.” By inspecting a large set of activations across diverse contexts, we may find that the feature’s top-activating examples are overwhelmingly associated with temporal expressions. This consistent association suggests that the feature’s role is to represent information about time.

Similarly, another feature $z_k^{(l)}$ might rank highest on inputs containing earnings-related language, such as “earnings per share,” “revenue,” or “earnings call.” Observing that its strongest activations almost exclusively occur in the presence of financial-reporting concepts leads us to label it as a corporate-earnings feature.

In this way, the meaning of each feature is inferred by collecting its highest-activation contexts, identifying the shared semantic pattern among them, and assigning a descriptive label that captures that pattern.¹²

D. Implementation

The sparse autoencoding approach introduced in the previous sections provides a lightweight and practical method for analyzing and manipulating LLMs. One of its core advantages is that it does not require retraining the LLM itself. Instead, the SAE is trained independently on the residual stream of a pre-trained model. This makes the method computationally efficient and accessible, requiring only a fraction of the data and computational resources typically needed to train or fine-tune a full-scale LLM.

In practice, the SAE can be viewed as an interpretable “plug-in” that augments the internal representations of a transformer model without altering its original parameters. This separation is particularly advantageous in settings where full retraining is impractical or undesirable, such as applied research in finance. Moreover, the increasing availability of open-source models and pretrained SAEs makes this approach highly suitable for academic and applied analysis.

Throughout this paper, we utilize the open-source Sparse Autoencoders released by Google DeepMind (Lieberum et al., 2024), trained on intermediate representations from the instruction-tuned language model Gemma-2-9B-IT. These models provide an ideal testbed for our study, combining open accessibility with strong performance on language understand-

¹²This is typically done by collecting the tokens that most strongly activate the feature and prompting a language model (e.g., GPT) to summarize the common concept or theme from these tokens.

ing tasks. For interpreting the sparse features, we use annotations provided by Neuronpedia,¹³ a crowd-sourced knowledge base that uses GPT-3.5-Turbo and GPT-4 to generate human-readable descriptions based on the top-activating tokens for each feature.

E. Concept Steering

The first benefit of SAE-augmented LLMs, discussed in Section IV, is the sparse representation itself, which enables the construction of *interpretable* text embeddings. These embeddings help understand how LLMs process financial documents. The second benefit is that labeled inner representations facilitate the possibility of pushing or *steering* LLMs toward specific concepts such as risk aversion, sentiment polarity, or other preferences and beliefs (Section V).

At a high level, steering can be understood as follows:

1. Identify a relevant feature in the sparse representation to steer the LLM toward. For instance, selecting one associated with risk to promote risk-averse behavior.
2. Use the SAE’s decoder to identify the perturbation induced by activating this feature; specifically, what would have been *added* to the residual stream if the feature were more strongly activated.
3. Add this perturbation to the residual stream to obtain a steered probability distribution over the next tokens.
4. Iteratively sample tokens from this steered distribution to generate text aligned with the intended concept (e.g., more risk-averse responses).

Formally, recall from Equation (2) that the probability of the next token in an LLM is obtained by applying the final block to the last residual stream:

$$P(\text{Next Word} \mid \text{Input Text}) = \text{Block}^{(\text{Output})}(\mathbf{r}^{(m)}).$$

Assume a SAE has been trained on the m^{th} residual stream, with encoder¹⁴

$$\mathbf{z}^{(m)} = f(\mathbf{W}_e \mathbf{r}^{(m)} + \mathbf{b}_e),$$

and decoder

$$\hat{\mathbf{r}}^{(m)} = g(\mathbf{W}_d \mathbf{z}^{(m)} + \mathbf{b}_d).$$

¹³See <https://www.neuronpedia.org/>

¹⁴In practice, SAE can be trained on any intermediary layer, but for the sake of clarity we focus on this simplified example. The same logic applies if the perturbation occurs earlier in the LLMs.

Suppose a relevant feature $z_j^{(m)}$ of the sparse representation has been identified—for example, one associated with positive sentiment. Steering then consists of generating a perturbation vector

$$\mathbf{e}_j = g(\mathbf{W}_d \mathbf{S}_j^{(m)} + \mathbf{b}_d),$$

where $\mathbf{S}_j^{(m)}$ is a steering vector that is zero everywhere except at the j^{th} coordinate, which is assigned a nonzero steering value $s \neq 0$:

$$\mathbf{S}_j^{(m)} = [0, \dots, 0, \underbrace{s}_{j^{th} \text{ position}}, 0, \dots, 0]. \quad (12)$$

The probability of the next token under steering is then

$$P^{(\text{Steered})}(\text{Next Word} \mid \text{Input Text}) = \text{Block}^{(\text{Output})}(\mathbf{r}^{(m)} + \mathbf{e}_j).$$

As we show in Section V.B, if $z_j^{(m)}$ is associated with positive sentiment, texts generated using $P^{(\text{Steered})}$ are likely to be more optimistic than those generated without steering. Conversely, steering toward risk-related features produces more cautious-sounding text.

This approach provides a fine-grained, concept-level control mechanism grounded in interpretable internal features. Unlike prompt engineering, which relies on heuristic text manipulations and often lacks transparency, steering via sparse features operates directly on the model’s latent representations. This control enables systematic and reproducible interventions without altering the model’s parameters or retraining.

III. Data

Throughout the paper, we use Reuters news articles related to individual U.S.-listed firms from 2015 to 2024, covering a ten-year period. We first follow the cleaning procedure described in Chen et al. (2022). Next, we reduce our sample to after-hours. Those two procedures result in a corpus of 3,664,197 articles, each consisting of a headline and a body. These news items are matched to firm returns from CRSP. Table I reports the corresponding summary statistics.

Sample and Look-Ahead Bias: A key concern in applying LLMs to finance is the potential look-ahead bias arising from the training sample (Sarkar and Vafa, 2024). A common approach to mitigate this issue is to restrict the sample to a relatively short period following the model’s training cutoff, thereby sacrificing statistical power to eliminate bias (Ludwig

Table I. Summary Statistics

This table reports summary statistics for the Reuters news dataset (2015–2024). For each year in the sample, we report the number of unique firms and news articles. We also provide the percentage of positive intraday returns, along with the average length of headlines and article bodies, measured in number of characters.

Year	Number of Firms	Number of News	% Return > 0	Text length	
				Headline length	Body length
2015	3,700	360,040	50.9%	87	1,215
2016	3,598	332,235	51.1%	87	1,266
2017	3,538	329,957	51.3%	88	1,209
2018	3,532	332,634	51.1%	84	1,203
2019	3,541	365,459	52.5%	85	1,176
2020	3,652	371,007	52.5%	86	1,120
2021	4,155	369,580	51.2%	85	1,138
2022	4,259	355,947	49.4%	85	1,114
2023	4,149	448,961	49.5%	79	1,140
2024	3,887	398,377	51.1%	76	1,299

et al., 2025). In this paper, however, we expand the sample to the last 10 years for two reasons.

First, the model employed, *Gemma-2-9B-IT*, is comparatively small by LLM standards. Smaller models are less likely to “memorize” specific events and thus exhibit reduced susceptibility to look-ahead bias. [Didisheim, Fraschini, and Somoza \(2025a\)](#) demonstrate that smaller OpenAI models showed virtually no ability to recall the returns of individual firms, even when prompted with information designed to “trigger” such memory, such as the corresponding market return on a given day.

Second, our analyses are either comparative-demonstrating that sparse embeddings outperform alternative embeddings derived from the same LLM (Sections [IV.A](#) and [V](#))-or centered on interpretability rather than predictive performance (Section [IV.B](#)).

Therefore, we argue that any residual look-ahead bias is likely minimal and does not affect the validity of our empirical claims.

IV. Interpreting the Embedding

In this section, we examine the economic performance and interpretability of the sparse embeddings produced by the SAEs (see Section [II](#)). Unlike simpler bag-of-words embeddings (see, e.g., [Tetlock, 2007](#)), SAE embeddings are derived from a pre-trained LLM and therefore capture subtle textual nuances that can be informative in a wide range of applications.

Moreover, in contrast to previously proposed LLM-based embeddings (Sarkar, 2025), SAE embeddings are explicitly designed to be interpretable.

A. Performance

Embeddings can be applied to various tasks, including firm similarity Sarkar and Vafa (2024) and ownership representation Gabaix, Koijen, Richmond, and Yogo (2024). We focus on portfolio construction, since this standard task has well-defined performance metrics and offers a natural setting to assess both the predictive power and interpretability of SAE embeddings.

Following the approach of Chen et al. (2022), we construct optimal portfolios as follows:

1. We use Gemma9B with augmented SAEs to generate sparse, interpretable vectors ($\nu_{i,t}$) for each news article in our sample (see Section III).
2. Using a rolling window of four years, we estimate a logistic regression model to predict the sign of subsequent intraday returns. The model is re-estimated annually.
3. We form a long-short intra-day portfolio that buys the 20% stocks with the highest average forecasted probability of positive return and shorts the 20% stocks with the lowest forecasted probability of positive returns.¹⁵

Since SAE models are generalists, each sparse feature is trained to capture only a limited subset of linguistic concepts. Consequently, the resulting embeddings are extremely high-dimensional, with many features irrelevant for financial applications. The model employed in this study produced embeddings with 131,000 features, spanning domains from finance to unrelated areas such as cooking. To address this dimensionality, we adopt the following procedure: we first train the model (as described above) on the first 1,000 principal components (PCs). We then propagate the PC loadings back to the original feature space, obtaining a ranking of features by the absolute magnitude of their loadings.¹⁶ We use this ranking to construct reduced embeddings consisting of the top k most relevant features. For a grid of k between 5 and 5,000, we repeat the training procedure described above to construct long-short portfolios. This procedure create a gird of portfolios constructed with predicting model relying on the top 5, 10, ..., 5,000 features. Table II reports the annualized Sharpe ratios and predictive accuracies for each k . The Sharpe ratio increases gradually from 3.34 for $k = 5$ to 5.51 for $k = 5,000$. The improvement appears to plateau around $k = 500$, which

¹⁵Days with fewer than 10 firms in either the long or short portfolio are excluded from the analysis.

¹⁶We re-estimate this ranking at every time step within our rolling window, thereby eliminating look-ahead bias.

already achieves a Sharpe ratio of 5.25. Similarly, forecasting accuracy rises from 50.49% for $k = 5$ to 51.55% for $k = 5,000$.

As a benchmark, we follow the methodology in [Chen et al. \(2022\)](#), who employ embeddings from the final layer of large language models and demonstrate that these representations outperform all prior textual features proposed in the finance literature. Importantly, their approach can be applied to extract embeddings from any LLM. This allows us to compute benchmark embeddings from the same LLMs used to generate the sparse embeddings, enabling a direct, like-for-like comparison. The portfolio constructed using this benchmark embedding as input to the logistic regression yields an annualized Sharpe ratio of 4.91. Using the test proposed by [Jobson and Korkie \(1981\)](#), with the correction suggested in [Mommel \(2003\)](#), we find that this difference is statistically significant at the 5% level (p -value = 0.042).

Taken together, these results yield two key insights. First, the interpretability of SAEs does not come at the expense of performance. On the contrary, they substantially outperform the recent methods presented in [Chen et al. \(2022\)](#). Second, while high performance can already be achieved with as few as five features, performance continues to improve as more features are incorporated. This pattern suggests that SAE embeddings exhibit a “virtue of complexity” in the sense of ([Kelly et al., 2024](#); [Didisheim, Ke, Kelly, and Malamud, 2024](#)).

Recall that SAE embeddings are sparse and semantically meaningful representations of an LLM’s text processing, in our case *Gemma9B*. Hence, this notion of a “virtue of complexity” extends to LLMs analyzing financial news more broadly. Indeed, the exercise in this section and the prompting-based sentiment estimation approach latter discussed in Section [V](#) or [Lopez-Lira and Tang \(2023a\)](#); [Chen et al. \(2024\)](#) are closely related. In both cases, we employ part of an LLM to compress news into a lower-dimensional representation. In the prompting approach, this representation is further processed by the LLM to generate text from which predictions are parsed. In the approach presented here, the lower-dimensional representation is instead mapped directly into an interpretable embedding, which is then used to predict returns.

In both methodologies, the prediction-and the subsequent portfolio construction-can be understood as a projection of these lower-dimensional representations. Consequently, the “virtue of complexity” observed in [Table II](#) does not merely reflect the complexity of the embeddings themselves, but extends to the underlying LLMs and their prompt-based applications.

Table II. Return Predictions and Number of Features

Out-of-sample performance of predictive models trained with varying reduced sparse representation as input. Each model produces news-level predicted probabilities, which are used to construct long-short portfolios based on the top and bottom 20% of predictions. “Benchmark” denotes the model trained with the embedding approach of [Chen et al. \(2022\)](#). Reported are the equal-weighted long-short Sharpe ratios, their statistical significance (p-values) relative to the model with 5000 features (with the alternative hypothesis that the Sharpe ratio is lower than that of the full embedding), as well as average daily accuracy (with next open-to-close returns as labels).

		Sparse features											
	Benchmark	5	10	30	50	100	300	500	1000	2000	3000	4000	5000
EW Sharpe	4.91 (0.042)	3.34 (0.000)	3.55 (0.000)	4.16 (0.000)	4.32 (0.000)	4.71 (0.000)	5.21 (0.035)	5.25 (0.089)	5.19 (0.019)	5.29 (0.065)	5.29 (0.071)	5.32 (0.072)	5.51
Accuracy	51.27%	50.49%	50.52%	50.85%	50.82%	51.12%	51.39%	51.44%	51.42%	51.52%	51.53%	51.47%	51.55%

B. Interpretability

Having shown in the previous section that our embedding does not sacrifice performance, we now turn to its main advantage over previous methods: interpretability. Each feature in our sparse embedding is labeled through the following procedure: (1) a large corpus of texts is processed by the LLM, (2) texts that “activate” a given feature (i.e., yield a non-zero value) are grouped together, and (3) another LLM is prompted to analyze these texts and assign an appropriate label. A detailed description of these steps is provided in Section II.

When the number of features is small enough, the assigned labels allow for direct interpretability. Figure 3 illustrates this by reporting the labels of the 5 most relevant features used in Table II. As a proxy for feature importance, the figure presents the absolute values of the model loadings from our portfolio exercise. Since we train one model per year, the reported value corresponds to the average loading across years.

The labels correspond to terms that one would reasonably expect when processing financial text to assess short-term revenue impact, such as “phrases indicating progress or improvement in performance” or “economic terms related to stock market performance and fluctuations.” This exercise can be interpreted as a validation of the methodology, as the top ten features exhibit coherent and intuitive labels. However, to rigorously analyze the main drivers behind LLMs’ interpretation of financial text, it is necessary to examine a broader set of features. In the remainder of this section, we propose a simple method to achieve this.

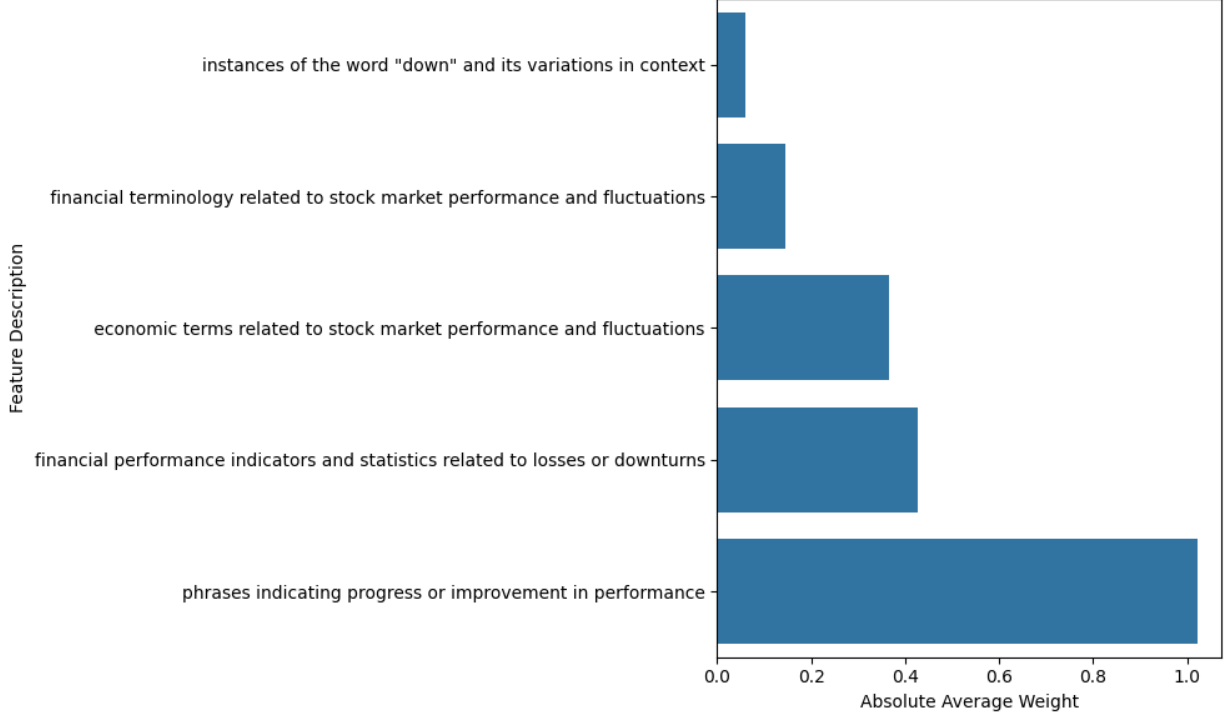


Figure 3. Top 5 Contributing Features

This figure reports the labels of the five features with the highest contribution to return prediction, obtained by replicating the exercise of [Chen et al. \(2022\)](#). For each feature, the bars report the absolute weights assigned by the forecasting model, averaged across rolling windows.

Our experiment in Section [IV.A](#), which uses 5,000 features as input and produces an out-of-sample score of 5.51, can therefore be interpreted as being mapped to 5,000 distinct feature labels. Since this number is too large for direct interpretation, we construct higher-level clusters of features using the following steps:

1. We embed the labels of the 5,000 features into numerical representations using the embedding model introduced by [Wang, Yang, Huang, Yang, Majumder, and Wei \(2023\)](#), a state-of-the-art model widely used for similar tasks.
2. We apply the k-means algorithm ([McQueen, 1967](#)) to cluster these embeddings into distinct groups. This algorithm iteratively updates centroids to minimize within-cluster variance, thereby grouping features with similar representations. The optimal number of clusters is determined by maximizing the silhouette coefficient ([Rousseeuw, 1987](#)), following standard practice in this literature.
3. These steps yield 25 clusters. However, several clusters exhibit highly similar economic interpretations. We therefore manually merge them, resulting in 17 unique clusters,

which we label with the following 17 concepts:¹⁷

Energy and Manufacturing	Finance/Corporate	Finance/Markets	Fixed Effects	Governance and Politics
Healthcare	Legal	Marketing & Retail	Others	Punctuation and Symbols
Quantitative	Risks	Scientific Jargon	Sentiment	Technical Analysis
Technology	Temporal Concepts			

Figure 4 presents the most frequent words appearing in the feature labels within each cluster. Several wordclouds, such as those for *Sentiment*, *Legal*, or *Healthcare*, exhibit relatively self-evident classifications. However, two clusters in particular merit further discussion.

The first is panel (f) of Figure 4, corresponding to the cluster labeled *Fixed Effects*. As the wordcloud indicates, this cluster centers on named entities such as individuals, companies, and locations. This suggests that the embeddings associated with this cluster are activated when specific entities appear in the prompt. In classical economics, the concept most closely related to firm specific variations is that of fixed effects, which motivates the cluster’s label. A related notion is look-ahead bias (Sarkar and Vafa, 2024), as well as the ability of LLMs to recall key economic values (Didisheim et al., 2025a).

The second cluster of interest is *Punctuation and Symbols*. The corresponding wordcloud, shown in panel (g) of Figure 4, displays words more strongly associated with coding and programming than with punctuation per se. This peculiarity arises from the feature labeling procedure. As described in Section II, labels are assigned by passing a large general-purpose textual dataset through the LLM and recording the types of texts or lines most frequently activating a given feature. Since a large share of code consists of punctuation (for example, most C++ lines end with “;” and colons initiate conditional or loop statements in Python), the labeling process mechanically associates the cluster with punctuation. We confirmed this interpretation by manually inspecting the texts linked to feature labeling.

We use these clusters to estimate prediction models based solely on the features within each cluster. In other words, we replicate the procedure described in Section IV.A, but instead of using all 5,000 features, we restrict the input to the subset belonging to a given cluster. This exercise produces 17 sets of out-of-sample forecasts, one for each cluster. Figure 5 reports the pairwise correlations among these forecasts.

The correlations are uniformly positive and generally high, ranging from 0.12 to 0.84. This result is expected: each forecast relies on a subset of the 5,000 most informative features extracted from sparse representations, all optimized to predict the same target variable—the sign of the 3-day cumulative returns. Moreover, these features reflect how LLMs encode information across different semantic concepts. LLMs represent relationships between tokens

¹⁷Appendix A transparently reports the choices made to merge the 25 clusters into 17 groups and provides the final naming of each group.



Figure 4. Wordclouds

These figures display the most frequent words in the feature labels within each cluster. The subcaptions indicate the names assigned to the respective clusters.

through the mechanism of *attention* (et al., 2017). Specifically, masked attention heads ensure that each token (word or subword) is embedded relative to preceding tokens. For instance, when processing the sentence “10% growth today,” the token “today” is encoded with its relationship to “growth” and “10%,” which likely contributes to the observed forecast correlations.

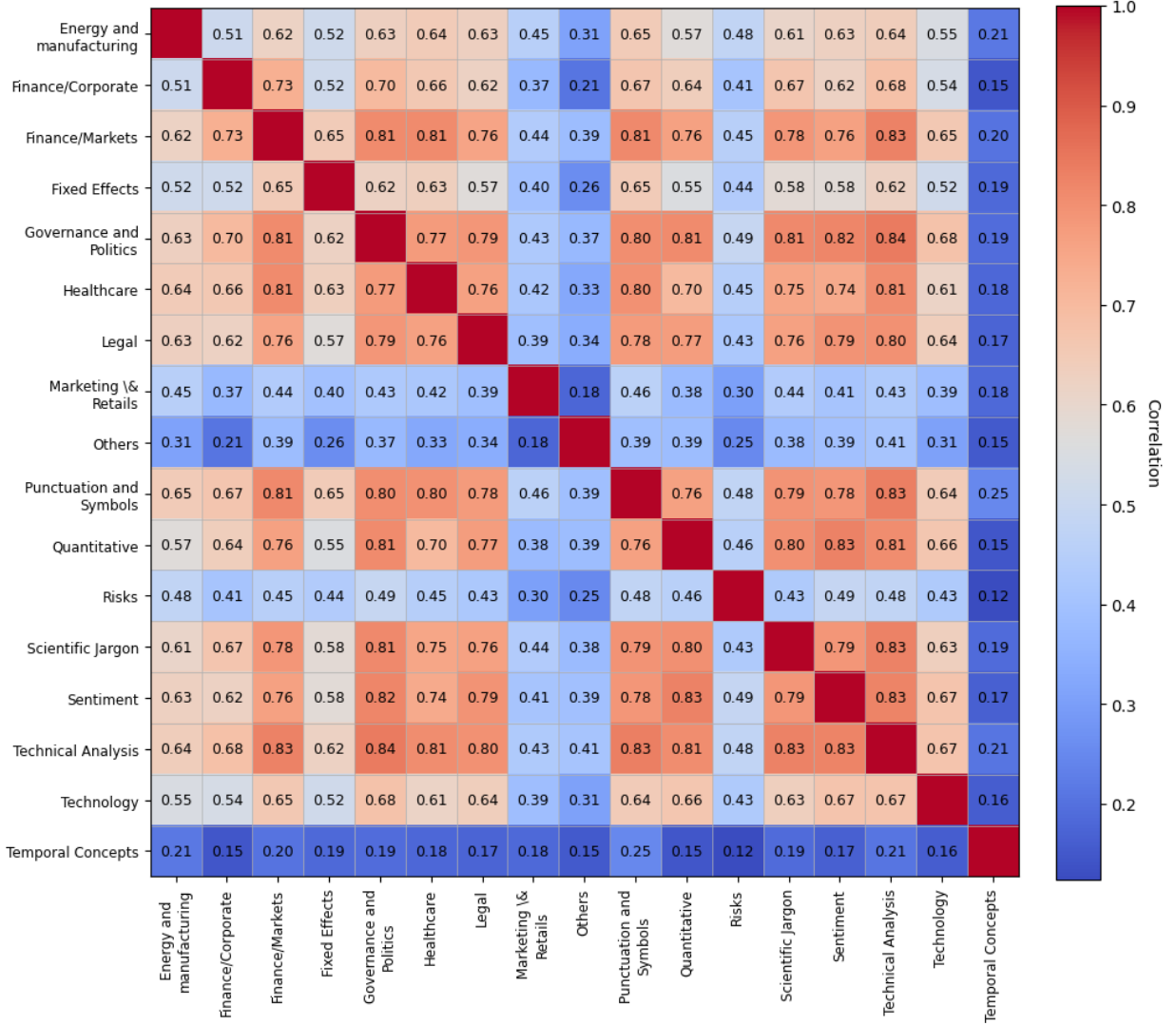


Figure 5. Correlation Between Predictions Across Clusters

Pairwise correlations of out-of-sample return predictions from models estimated on individual feature clusters. Blue indicates lower correlations, whereas red denotes higher correlations.

We next construct an alternative set of predictions in the spirit of Shapley values (Gu

et al., 2020). For each labeled cluster (e.g., Sentiment, Risks), we form a feature set consisting of all 5,000 features excluding those belonging to that cluster. We then apply the procedure described in Section IV.A to obtain prediction accuracies and the Sharpe ratios of the corresponding long-short strategies.

Table III reports the *Shapley Sharpe*, defined as the Sharpe ratio obtained using the full set of 5,000 features (5.51) minus the Sharpe ratio obtained when excluding the features of a given cluster ($SR_{5,000} - SR_k$ where SR_k is the Sharpe ratios obtained with k features). The second column reports the Sharpe ratio obtained when using *only* the features in that cluster to construct the prediction model. For both values, we report the statistical significance of their difference relative to the full Sharpe ratio (5.51) (Jobson and Korkie, 1981; Memmel, 2003). These two columns provide complementary insights. The Shapley Sharpe captures the incremental predictive power unique to a given cluster, in the sense that the logistic predictor cannot replicate the same information from correlated features in other clusters. By contrast, the individual Sharpe reflects the stand-alone predictive power of the cluster.

Several findings emerge. First, all individual Sharpes are significantly below the full Sharpe, but not all Shapley Sharpes are significant. Thus, no single cluster suffices for maximum performance, and some are unnecessary.

Second, examining the ranking of topics by Shapley Sharpe reveals that the top three clusters—*Sentiment*, *Finance/Markets*, and *Technical Analysis*—capture fundamental concepts in finance and portfolio construction as reflected in news. Recall that the SAE embeddings under analysis provide an interpretable low-dimensional representation of an LLM’s internal state when processing text. They can thus be interpreted as an approximation of the LLM’s reasoning process. It is therefore unsurprising that the three most influential clusters reflect concepts central to financial news interpretation.

The next most important cluster, however, is less expected: temporal concepts. This cluster exhibits a statistically significant Shapley Sharpe of 0.41, yet one of the lowest individual Sharpe ratios (1.27). This apparent discrepancy becomes clearer when recognizing that the cluster encodes notions of timing. Our application focuses on predicting short-term news-driven returns. The dominant driver of such predictions is the directional content of news—whether it conveys positive or negative information about a company. Clusters such as *Sentiment*, *Finance/Markets*, and *Technical Analysis* directly capture this dimension. The secondary determinant is the timeframe—whether the news is likely to affect markets in the short or long run. Timing features thus add value only in conjunction with sentiment-related features, which is precisely what Table III reflects. Panel (d) of Figure 4 confirms this interpretation, showing that this cluster indeed captures timing notions with key terms such as

“dates,” “particular event,” and “timelines.”

The cluster *Punctuation and Symbols* exhibits the opposite pattern. It displays no significant Shapley Sharpe but a very high individual Sharpe of 4.65. This result is consistent with the way transformers encode relationships between words and symbols. As noted earlier, the *attention* mechanism ensures that words are represented in relation to their surrounding context. Symbols such as “.” or “,” carry little predictive power in isolation, but LLMs likely encode them jointly with semantic content. Consequently, the logistic classifier can, for example, distinguish a positive period at the end of a positive sentence from a negative period at the end of a negative one. This explains the observed pattern: punctuation and symbol features have little unique predictive value but provide a weak aggregation of contributions from other clusters.

The *Quantitative* cluster also warrants attention. It does not yield a significant Shapley Sharpe and produces below-median individual Sharpe. This finding aligns with evidence that LLMs struggle with quantitative reasoning (Liu, Wu, Wu, Lu, Chang, and Feng, 2024). In finance, where performance often requires a combination of quantitative and qualitative analysis, this limitation is particularly salient. Consistent with prior work (Didisheim, Fraschini, Somoza, and Tian (2025b)), our results indicate that LLMs rely exclusively on qualitative reasoning when processing financial news.

Another relevant cluster is *Fixed Effects*, which captures both firm-specific or entity-specific encodings in the econometric sense, and may also reflect the LLM’s propensity for look-ahead bias (Sarkar and Vafa, 2024). While look-ahead bias is unlikely to be confined entirely to this cluster, it is plausible that tokens tied to specific firms, individuals, or events contribute disproportionately. The modest but statistically significant Shapley Sharpe for this cluster (at the 5% level) is consistent with recent findings that look-ahead bias is relatively limited in portfolio applications, especially when using smaller LLMs as in this study (Glasserman and Lin, 2023; Didisheim et al., 2025a).

Finally, we find no evidence of correlation between the number of features in a cluster (reported in the last column of Table III) and its importance. This suggests that the clusters capture distinct and meaningful dimensions of information. A random allocation of features across clusters would likely yield a positive relationship, with larger clusters exhibiting stronger performance. The absence of such a relationship supports the interpretability and substantive relevance of the clustering.

Table III. Return Predictions and Topics

Out-of-sample performance of predictive models using different topic groupings during training. Topics are constructed via K-means clustering on embedded feature descriptions. The table reports: (i) *Raw Sharpe*, the performance of models trained on features from a single topic only; and (ii) *Marginal contribution (LOFO Shapely values)*, computed by re-training the full model while leaving one feature out (Leave-One-Feature-Out). The performance drop relative to the full-feature model reflects that feature’s marginal contribution, which is then summarized as a Shapely value.

	Shapley Sharpe	Individual Sharpe	Avg Daily Accuracy	Total Accuracy	Num features
ALL		5.51	51.55%	51.44%	5000
Sentiment	0.54***	3.81***	50.69%	50.47%	240
Finance/Markets	0.42**	4.89***	51.30%	51.28%	399
Technical Analysis	0.41***	4.82***	51.19%	51.09%	431
Temporal Concepts	0.41***	1.27***	49.81%	49.89%	71
Risks	0.41**	4.80**	50.87%	50.71%	97
Finance/Corporate	0.38**	5.14*	51.23%	51.09%	164
Marketing & Retails	0.33**	3.76***	50.55%	50.59%	111
Technology	0.30*	4.24***	51.02%	50.95%	252
Fixed Effects	0.29**	4.54***	51.01%	51.00%	176
Scientific Jargon	0.28*	4.93**	50.96%	50.91%	226
Healthcare	0.28*	4.55***	51.12%	51.12%	295
Governance and Politics	0.26*	4.78**	51.27%	51.12%	764
Energy and manufacturing	0.18	4.82**	51.01%	51.06%	102
Others	0.17	2.47***	50.04%	49.86%	75
Punctuation and Symbols	0.07	4.65***	51.17%	51.09%	762
Quantitative	0.05	3.93***	50.82%	50.62%	582
Legal	0.05	3.87***	50.85%	50.74%	253

V. Steering LLM Sentiment

We now turn to applications of *steering*, discussed in detail in Section II.E. In essence, the approach consists of identifying relevant concepts in the SAE’s representation (e.g., risk aversion or optimism/pessimism) and selectively amplifying or attenuating them during text generation, thereby producing outputs *steered* toward those concepts. This approach is fundamentally different from prompt engineering.

Consider the example of risk aversion. A prompt-engineering strategy would involve modifying the prompt to explicitly instruct the LLM to behave cautiously, or adding keywords such as “risk” or “danger.” Such methods rely on trial-and-error and must often be tailored to each specific task. Furthermore this method is unable to surgically alter a given concepts.

By contrast, steering identifies the feature in the sparse representation associated with risk and leverages it to activate all neurons in the LLM linked to that concept. Rather than adjusting prompts individually, steering alters the generative process itself by selectively enhancing or suppressing neurons associated with a given concept leaving all else equal. This makes the approach broadly applicable across tasks, since the same steer can be reused without repeated trial-and-error. Moreover, it provides an interpretable scale: while two degrees of prompt engineering cannot be objectively compared in terms of “risk aversion,” the strength of steering is explicitly controlled by a continuous parameter.

Before turning to the next section, where we examine how steering can be applied to de-bias LLMs in financial contexts, we first present a simple “proof of concept” to test whether steering produces the expected outcome in an economic classification task. Using the news dataset described in Section III, we prompt the LLM to classify each news item as positive or negative with the following instruction:

Here is a news headline: {headline}.

Here is the news body: {body}.

You are a financial news analyst that answers (with no further explanation) either P (positive return) or N (negative return) for the news given for a company. Please answer in a single letter.

Prompt 1. News Classification

The prompt asks the LLM to classify a news item as either positive or negative. The prompt is used for all the exercises presented in this section.

We then repeat the procedure while *steering* the model: by intervening on a node linked to positivity/optimism, we steer the node in either the positive or negative direction.¹⁸ This generates a set of predictions in which, for each news item, we obtain classifications under a grid of positive and negative steering levels, alongside a benchmark with no steering. As in Section IV, we use the next day’s intraday return as the classification label; that is, each news item is paired with the corresponding firm’s realized return.

Table IV reports the results. The proportion of positive classifications increases monotonically with the steering coefficient. When the model’s positivity node is activated, it is more likely to predict a positive market reaction. The corresponding return patterns are consistent with this behavior: under negative steering, positive classifications are associated higher next day average returns, whereas under positive steering, negative classifications are associated with lower next day average returns. These findings support the conclusion that the steering mechanism operates as intended.

¹⁸Index 111712 (expressions of positive sentiment and appreciation) on google/gemma-scope-9b-it-res (layer 20) with 131k features.

Table IV. Validating the Steering Technique

This table reports the results of providing the model with individual news items and asking it to classify each as positive or negative for the firm. *Steering* denotes the degree of manual activation of the LLM feature associated with positivity. *Positive Class.* indicates the percentage of news items classified as positive. *Average Return* corresponds to the mean daily stock return on the news day, reported for all observations and separately for positive and negative classifications.

Steering	Positive Class.	Average Return	
		Positive	Negative
-100	56.0%	0.0145	-0.0105
-50	58.7%	0.0141	-0.0115
-30	61.0%	0.0136	-0.0123
0	64.5%	0.0130	-0.0137
30	67.5%	0.0124	-0.0150
50	69.6%	0.0121	-0.0161
100	77.2%	0.0108	-0.0210

A. Optimism Bias and LLMs

A common feature of LLM outputs is a perceived bias toward positive sentiment, potentially driven by marketing considerations (see, e.g., [Fatahi et al., 2024](#)). To test for this bias in LLM-based financial analysis, we conduct a portfolio study using the classifications obtained above for different steering coefficients.

Following [Lopez-Lira and Tang \(2023a\)](#), we construct daily long-short, dollar-neutral portfolios based on news classifications, relying exclusively on after-hours news and trading on the subsequent day.

Figure 6, panel (a), reports the annualized Sharpe ratios for each steering coefficient, including the 0-steering baseline. To assess significance, panel (b) presents the annualized alpha from the regression

$$r_t^{\text{Steering}=s} = \alpha + \beta r_t^{\text{Steering}=0} + \varepsilon_t,$$

where $r_t^{\text{Steering}=s}$ is the return of the long-short portfolio with steering coefficient s .

The results indicate a clear positive bias. The baseline Sharpe ratio equals 3.87, while negatively steered portfolios systematically achieve higher ratios (4.07, 4.11, and 4.28). In contrast, positively steered portfolios yield ratios that are either comparable or substantially lower (3.90, 3.74, and 2.71). Panel (b) further shows that the difference between negatively

steered coefficients and the baseline is statistically significant.

These findings validate steering as a tool to both identify and mitigate LLM bias. By selecting an appropriate steering level for a given feature, one can correct biases such as those documented in Fedyk et al. (2024). Conversely, steering can also be employed to deliberately amplify bias for research purposes—for instance, to simulate varying levels of risk aversion or home bias in human decision-making. We discuss this broader methodological potential in Section V.B.

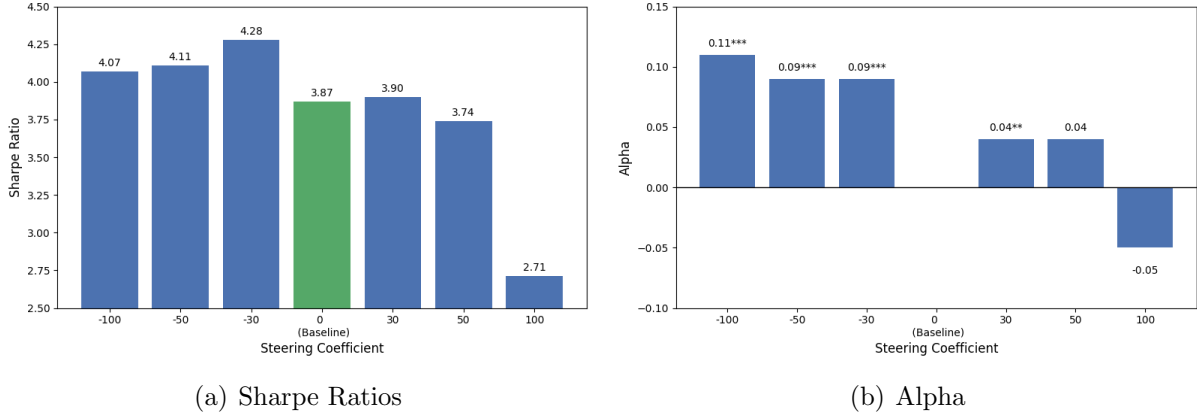


Figure 6. Comparison of model performance metrics. (a) Sharpe ratios for the benchmark and different steering levels. (b) Alpha estimates for steering levels, with significance denoted by stars.

B. General Use of the Methodology

The LLM steering technique is adaptable across the social sciences to study diverse behaviors and decision-making patterns. By identifying and manipulating interpretable the internal features of an LLM, researchers can simulate agents with specific dispositions, preferences, and contextual sensitivities. For example, they can adjust a model’s implied risk aversion to assess its impact on economic choices or guide it to emphasize social, political, or cultural dimensions relevant to a research question. Because the framework operates on internal representations, these adjustments can target specific conceptual traits while holding other factors constant. This allows controlled experiments that extend beyond finance to psychology, sociology, political science, and other fields concerned with decision processes. In this section, we provide a few examples.

We start with an exercise where we ask the LLM to make a an investment decision in two different context. With the first prompt, we ask it to make a portfolio choice where it has

to allocate 100 dollars between two asset classes: treasuries and the S&P500. The prompt is the following:

This is a simulation for a university assignment.

You have \$100 to split between two options:

1. US Treasuries - low risk and low return.
2. An ETF tracking the S&P500 - higher risk but potentially higher return.

This is just a simulation. You are not giving any financial advice, and nobody will invest anything based on your answer.

How much do you put in each option?

Do not explain your choice, simply state it.

Prompt 2. Risk Aversion and Portfolio Choice

This prompt asks the LLM to allocate \$100 between two asset classes with different risk levels. We use it in an exercise where we manually steer the model's risk aversion and assess whether the allocation matches the expected outcome.

With the second prompt, we ask it to invest in either a startup or a fixed income fund. We use the following prompt:

This is a simulation for a student finance project.

You have \$100 and must choose one of two options:

1. A high-growth tech startup - high risk, high return.
2. A municipal bond fund - low risk, low return.

This is just a simulation. You are not giving any financial advice, and nobody will invest anything based on your answer.

Which option do you choose?

Do not explain your choice, simply state 1 or 2.

Prompt 3. Risk Aversion and Investment Choice

This prompt asks the LLM to invest in either a startup or a fixed-income fund. We use it in an exercise where we manually steer the model’s risk aversion and assess whether the allocation matches the expected outcome.

For this exercise, we steer towards risk aversion and wealth. For each concept, high and low directions are induced by modifying internal representations of the appropriate feature.

Each prompt is repeated 100 times to reduce the noise generated by the decoding strategy within the LLM. Table [V](#) reports the results.

Table V. Risk Aversion and Investment Choice

This table reports the results of querying the LLM 100 times under varying levels of risk aversion and attention to wealth, using Prompt 2 and Prompt 3. For Prompt 2, the values correspond to the dollar amounts allocated to the S&P500. For Prompt 3, the values indicate the chosen investment, where 1 denotes a startup and 2 a fixed-income fund.

Steered feature:	% allocated to S&P500			1=startup, 2=bond		
	financial performance	financial risk	wealth success	financial performance	financial risk	wealth success
Steering level						
-100	27.20	47.50	29.15	1.37	1.00	1.16
-75	28.20	39.05	29.95	1.37	1.00	1.15
-50	29.90	39.40	30.45	1.29	1.00	1.11
-25	32.10	40.00	31.95	1.15	1.00	1.07
0	38.45	38.45	38.45	1.02	1.02	1.02
25	44.35	32.80	47.55	1.00	1.03	1.00
50	46.40	30.80	49.25	1.00	1.11	1.00
75	49.70	29.80	50.50	1.00	1.15	1.00
100	54.95	29.55	50.25	1.00	1.28	1.00

VI. Conclusion

This paper presents a simple, scalable approach to opening and controlling large language models by inserting a sparse, interpretable layer between the model’s hidden states and its outputs. Training a sparse autoencoder on the residual stream produces features aligned with familiar economic concepts such as positivity, risk aversion, and attention to wealth. These features can be adjusted to influence the model’s reasoning while keeping the base weights fixed. This method bridges black-box performance and transparency.

Applied to firm-level news, concept steering performs as intended. Increasing a “positivity” feature raises the share of positive classifications and shifts return patterns in line with more optimistic interpretations of the same news item. Leveraging this method, we find that off-the-shelf LLMs are overly optimistic for trading applications. Reducing positivity improves portfolio performance, with the highest Sharpe ratio at a modest negative steering level and statistically significant gains over the unsteered benchmark.

This method also help us clarifying the drivers of short-horizon predictability. Replicating embedding-based return-forecasting (Chen et al., 2022) with interpretable features shows that a small set of sentiment-like features explains most of the signal, even under aggressive dimensionality reduction. This connects strong results in prior work to a transparent mechanism: news embeddings are effective largely because the model captures fine-grained sentiment.

The framework extends beyond finance. Steering features linked to risk aversion and wealth influence decisions in controlled prompts in the expected direction, showing that concept-level control can simulate agents with configurable preferences. This flexibility supports applications in political economy, psychology, and sociology, where controlled variation in dispositions is useful and retraining is costly.

While this technique is still in its infancy, concept steering could evolve into a standard tool for building interpretable, field-ready LLMs for research and practice with countless applications across all social sciences.

REFERENCES

- Aguilera, Ana M, Manuel Escabias, and Mariano J Valderrama, 2006, Using principal components for estimating logistic regression with high-dimensional multicollinear data, *Computational Statistics & Data Analysis* 50, 1905–1924.
- Aher, Gati, Rosa I. Arriaga, and Adam Tauman Kalai, 2023, Using large language models to simulate multiple humans and replicate human subject studies, in *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research* (PMLR).
- Argyle, Lisa P., Ethan C. Busby, Nancy Fulda, Joshua R. Gubler, Christopher Rytting, and David Wingate, 2023, Out of one, many: Using language models to simulate human samples, *Political Analysis* 31, 337–351.
- Bybee, Leland, Bryan Kelly, Asaf Manela, and Dacheng Xiu, 2024, Business news and business cycles, *The Journal of Finance* 79, 3105–3147.
- Bybee, Leland, Bryan T. Kelly, Asaf Manela, and Dacheng Xiu, 2020, The structure of economic news, SSRN Working Paper No. 3522305.
- Chen, Hui, Antoine Didisheim, and Luciano Somoza, 2024, Out of the (black) box: Ai as conditional probability, *Available at SSRN* .
- Chen, Yifei, Bryan T. Kelly, and Dacheng Xiu, 2022, Expected returns and large language models, SSRN Working Paper No. 4543999.
- Cheng, Qiang, Pengkai Lin, and Yue Zhao, 2024, Does generative ai facilitate investor trading? evidence from chatgpt outages, *Available at SSRN* 4872189.
- Cunningham, Hoagy, et al., 2023, Sparse autoencoders find highly interpretable features in language models, *arXiv preprint arXiv:2309.08600* .
- Didisheim, Antoine, Martina Fraschini, and Luciano Somoza, 2025a, Ai’s predictable memory and its implications for finance, SSRN Working Paper No. 5331177.
- Didisheim, Antoine, Martina Fraschini, Luciano Somoza, and Hanqing Tian, 2025b, Is AI reasoning useful in finance?, Unpublished Working Paper.
- Didisheim, Antoine, Shikun Barry Ke, Bryan T Kelly, and Semyon Malamud, 2024, Apt or “aipt”? the surprising dominance of large factor models, Technical report, National Bureau of Economic Research.

- et al., Vaswani, 2017, Attention is all you need, *Advances in neural information processing systems* 30.
- Fatahi, Somayeh, Julita Vassileva, and Chanchal K Roy, 2024, Comparing emotions in chatgpt answers and human answers to the coding questions on stack overflow, *Frontiers in Artificial Intelligence* 7, 1393903.
- Fedyk, Anastassia, Ali Kakhbod, Peiyao Li, and Ulrike Malmendier, 2024, Chatgpt and perception biases in investments: An experimental study, Available at SSRN 4787249.
- Gabaix, Xavier, Ralph SJ Koijen, Robert Richmond, and Motohiro Yogo, 2024, Asset embeddings, *Available at SSRN 4507511* .
- Gao, Leo et al., 2024, Scaling and evaluating sparse autoencoders, *arXiv preprint arXiv:2406.04093* .
- Gemini, 2025, Gemini: A family of highly capable multimodal models.
- Gentzkow, Matthew, Bryan Kelly, and Matt Taddy, 2019, Text as data, *Journal of Economic Literature* 57, 535–574.
- Glasserman, Paul, and Caden Lin, 2023, Assessing look-ahead bias in stock return predictions generated by gpt sentiment analysis, arXiv preprint arXiv:2309.17322.
- Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio, 2016, *Deep learning*, volume 1 (MIT press Cambridge).
- Gu, Shihao, Bryan T. Kelly, and Dacheng Xiu, 2020, Empirical asset pricing via machine learning, *Review of Financial Studies* 33, 2223–2273.
- Gu, Shihao, Bryan T. Kelly, and Dacheng Xiu, 2021, Autoencoder asset pricing models, *Journal of Econometrics* 222, 429–450.
- Horton, John J., 2023, Large language models as simulated economic agents: What can we learn from homo silicus?, NBER Working Paper 31122, National Bureau of Economic Research.
- Jobson, J Dave, and Bob M Korkie, 1981, Performance hypothesis testing with the sharpe and treynor measures, *Journal of Finance* 889–908.
- Kelly, Bryan, Semyon Malamud, and Kangying Zhou, 2024, The virtue of complexity in return prediction, *The Journal of Finance* 79, 459–503.

- Lieberum, Tom, et al., 2024, Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, *arXiv preprint arXiv:2408.05147* .
- Liu, Xiao, Zirui Wu, Xueqing Wu, Pan Lu, Kai-Wei Chang, and Yansong Feng, 2024, Are llms capable of data-based statistical and causal reasoning? benchmarking advanced quantitative reasoning with data, *arXiv preprint arXiv:2402.17644* .
- Llama, 2024, The llama 3 herd of models.
- Lopez-Lira, Alejandro, and Yuehua Tang, 2023a, Can chatgpt forecast stock price movements? return predictability and large language models, *arXiv preprint arXiv:2304.07619*.
- Lopez-Lira, Alejandro, and Yuehua Tang, 2023b, Can chatgpt forecast stock price movements? return predictability and large language models, *arXiv preprint arXiv:2304.07619* .
- Loughran, Tim, and Bill McDonald, 2011, When is a liability not a liability? textual analysis, dictionaries, and 10-ks, *The Journal of finance* 66, 35–65.
- Ludwig, Jens, Sendhil Mullainathan, and Ashesh Rambachan, 2025, Large language models: An applied econometric framework, Technical report, National Bureau of Economic Research.
- McQueen, James B, 1967, Some methods of classification and analysis of multivariate observations, in *Proc. of 5th Berkeley Symposium on Math. Stat. and Prob.*, 281–297.
- Mommel, Christoph, 2003, Performance hypothesis testing with the sharpe ratio, Available at SSRN 412588.
- OpenAI, 2024, Gpt-4 technical report.
- Park, Joon Sung, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein, 2023, Generative agents: Interactive simulacra of human behavior, in *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST ’23)* (ACM).
- Radford, Alec, et al., 2019, Language models are unsupervised multitask learners .
- Rousseeuw, Peter J, 1987, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, *Journal of computational and applied mathematics* 20, 53–65.
- Sarkar, Suproteem K, 2025, Economic representations.

- Sarkar, Suproteem K., and Keyon Vafa, 2024, Lookahead bias in pretrained language models, SSRN Working Paper No. 4754678.
- Shi, Wei et al., 2025, Route sparse autoencoder to interpret large language models, *arXiv preprint arXiv:2503.08200* .
- Tetlock, Paul C, 2007, Giving content to investor sentiment: The role of media in the stock market, *The Journal of finance* 62, 1139–1168.
- Vaswani, Ashish, et al., 2017, Attention is all you need.
- Wang, Liang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei, 2023, Improving text embeddings with large language models, *arXiv preprint arXiv:2401.00368* .

Appendix A. Topic Merging

Section IV.B describes the use of the k-means algorithm (McQueen, 1967) to cluster label features into 25 distinct groups. The choice of 25 clusters is guided by the maximization of the silhouette coefficient (Rousseeuw, 1987), consistent with standard practice in the literature.

To assign economic interpretations to the clusters, we examine the labels closest to each cluster centroid and apply economic judgment. Figures 7 to 11 display the ten labels nearest to the center of each cluster.

As discussed in Section IV.B, when clusters were difficult to interpret, we consulted the documentation available at <https://www.neuronpedia.org> to identify the types of texts used to generate the reported labels. For instance, although the labels in Topic 0 appear related to programming, inspection of the source texts revealed that this association arises because punctuation symbols-frequently occurring in code-were prominent in the labeling process.

The results yielded the following classification:

- 0, 3, 11, 22 → Punctuation and Symbols
- 1, 18 → Healthcare
- 2 → Sentiment
- 4 → Finance/Markets
- 6 → Technology
- 7 → Temporal Concepts
- 8 → Scientific Jargon
- 9 → Finance/Corporate
- 12 → Legal
- 5, 13 → Technical Analysis
- 24 → Risks
- 14, 17, 19 → Governance and Politics
- 15 → Energy and Manufacturing
- 10, 20 → Quantitative
- 16 → Fixed Effects
- 21 → Others
- 23 → Marketing & Retails

Rank	Feature
1	programming functions structures
2	programming code structures
3	programming code syntax structures
4	programming syntax data handling
5	programming functions definitions
6	programming syntax variable definitions
7	programming syntax structure
8	programming syntax components
9	programming coding structure
10	programming data structure definitions

(a) Topic 0

Rank	Feature
1	medical treatment research outcomes
2	medical conditions treatments
3	medical conditions treatments
4	medical treatments data
5	medical procedures outcomes
6	medical health impacts
7	medical health entities
8	medical technical measurements treatments
9	medical conditions health risks
10	medical assessments treatments

(b) Topic 1

Rank	Feature
1	negative sentiments conditions acceptance limitations
2	statements expressing uncertainty suggestion
3	statements regarding personal experiences opinions
4	statements doubt questioning certainty
5	expressing opinions judgments performance events
6	questions statements personal experiences desires
7	subjective assessments statements regarding conditions values
8	negative contradictory statements implications
9	awareness perception subjective evaluation conversations
10	opinions assessments events situations

(c) Topic 2

Rank	Feature
1	structures labels data organization representation
2	patterns structured data programming syntax
3	patterns data handling organization documents
4	patterns data structures representations
5	identifiers data structures machine learning datasets properties
6	structured data function definitions programming
7	structured data mathematical representations
8	structured data representations formats texts
9	patterns structured data likely programming technical
10	numerical identifiers metadata structured data formats

(d) Topic 3

Rank	Feature
1	financial values economic figures
2	financial figures investment
3	financial metrics trends stock markets
4	financial metrics statistics
5	financial metrics trends market performance
6	financial data investment opportunities
7	financial market analysis performance metrics
8	financial transactions economic activities
9	financial transactions market activities
10	financial statistics economic metrics industries

(e) Topic 4

Rank	Feature
1	forms
2	forms
3	analysis
4	review
5	engagement
6	article
7	increase
8	update
9	company
10	industry

(f) Topic 5

Figure 7. Top 10 features (by proximity) - Topics 0–5

Rank	Feature
1	information statistics performance sports
2	technology companies interactions
3	companies products relation social media dynamics
4	updates information financial performance corporate communications
5	companies organizations involved engineering technology
6	announcements new developments products events
7	online content news articles
8	social media platforms business dynamics
9	events actions competition ranking
10	discussions digital content subscription services

(a) Topic 6

Rank	Feature
1	dates time periods
2	time dates events
3	dates times mentioned document
4	dates times
5	dates times
6	dates time identifiers
7	dates months
8	dates numerical information
9	dates times sports
10	time events sequences

(b) Topic 7

Rank	Feature
1	scientific technical materials methods biological processes
2	scientific technical processes measurements
3	scientific technical measurements models
4	scientific biological research analysis
5	scientific technical measurements experiments
6	scientific biological measurements responses
7	scientific technical measurements processes
8	scientific research measurements values
9	scientific technical data analysis relationships
10	scientific biological processes experimental procedures

(c) Topic 8

Rank	Feature
1	contracts values outcomes
2	business transactions acquisitions
3	contractual agreements financial transactions
4	monetary values associations
5	rates financial
6	contracts negotiations
7	information payment processing systems fees
8	payment salaries financial arrangements
9	costs pricing products services
10	billing credits financial transactions

(d) Topic 9

Rank	Feature
1	mathematical operations equations variables
2	mathematical symbols structures
3	mathematical symbols equations
4	mathematical symbols equations
5	mathematical operations analysis
6	mathematical signs properties
7	mathematical symbols functions equations
8	mathematical operations
9	mathematical operations
10	mathematical operations

(e) Topic 10

Rank	Feature
1	indicative structure function technical
2	pronouns entities conflict resolution
3	punctuation marks symbols focus structure sentences
4	punctuation marks frequency text
5	indicative verbs changes trends conditions statistics
6	punctuation special characters quotes sentences
7	punctuations formatting indicate sentence structure flow
8	punctuation numerical information text
9	punctuation formatting characters
10	punctuation marks contextual cues used structure complex sentences

(f) Topic 11

Figure 8. Top 10 features (by proximity) - Topics 6–11

Rank	Feature	Rank	Feature
1	legal actions decisions	1	actions processes growth introduction assessment
2	legal proceedings contracts	2	actions processes measurement evaluation
3	legal processes arguments	3	conditions requirements decision making
4	legal proceedings opinions	4	actions processes research development
5	legal proceedings discussions	5	actions outcomes competition performance
6	legal court proceedings	6	patterns improvement growth metrics
7	legal court proceedings legal actions	7	evaluation assessment qualities conditions
8	legal court proceedings judgments	8	discussions choices implications particularly data project management
9	legal court cases proceedings	9	conditions results actions events
10	legal documents proceedings	10	analysis assessment processes

(a) Topic 12

(b) Topic 13

Rank	Feature	Rank	Feature
1	relationships connections subjects particularly testimony evidence	1	energy organizations activities
2	relationships actions features	2	transportation systems services
3	people organizations roles actions narrative	3	energy management transactions
4	people places events conflict societal issues	4	information manufacturing facilities production details
5	personal relationships social connections	5	details engineering manufacturing processes
6	social dynamics personal interactions	6	construction entities activities
7	individuals relationships	7	energy utility companies
8	themes communication collaboration individuals	8	topics construction infrastructure development regulatory achievements industries
9	social relationships roles individuals	9	energy resources economic impact
10	roles relationships interaction events	10	construction programming execution processes

(c) Topic 14

(d) Topic 15

Rank	Feature	Rank	Feature
1	names relevant individuals organizations locations	1	art media organizational structures
2	proper names titles individuals organizations	2	switzerland entities characteristics
3	names titles individuals involved legal sports	3	exchanges interactions
4	names titles organizations data sources	4	insiders official statements
5	names entities particularly proper titles	5	media control manipulation
6	names organizations companies	6	themes surveillance privacy infringements
7	names organizations associations	7	scandals political controversies
8	titles names organizations members	8	seattle entities
9	proper names individuals organizations	9	humor weekly activities
10	names entities organizations places	10	geographical locations

(e) Topic 16

(f) Topic 17

Figure 9. Top 10 features (by proximity) - Topics 12–17

Rank	Feature
1	biochemical processes modifications
2	biochemical compounds effects
3	biochemical processes molecular biology
4	biochemical processes molecular biology
5	scientific parameters molecular biology nanotechnology
6	biochemical physiological processes
7	biochemical processes brain function neuropharmacology
8	scientific vocabulary dosage concentrations experimental
9	biological biochemical products
10	information cancer studies tumor characteristics

(a) Topic 18

Rank	Feature
1	corporate entities interactions government religious organizations
2	political leaders organizations involved governance legal processes
3	government entities legal matters
4	organizations operational structures achievements
5	organizations entities government institutions
6	organizational governmental entities infrastructure projects
7	organizations entities systems
8	organizations official entities
9	government political processes
10	organizations documents parts formal reports

(b) Topic 19

Rank	Feature
1	numerical values statistics measurements
2	numerical values statistics measurements
3	numerical values statistics data
4	numerical data statistics quantities measurements
5	numerical data statistics measurements
6	numerical data statistics measurements
7	numerical data statistics measurements
8	numerical data statistics measurements
9	numerical data statistics measurements quantities
10	numerical data measurements statistics

(c) Topic 20

Rank	Feature
1	letter
2	letter
3	letter
4	letter
5	snap
6	note
7	proper
8	place
9	certain
10	miss

(d) Topic 21

Rank	Feature
1	technical data analysis programming
2	technical programming operations
3	technical data processing
4	technical scientific processes measurements
5	technical scientific processes measurements
6	technical data processing analysis
7	technical structures programming data management
8	technical programming data analysis
9	technical components systems
10	technical parameters programming data analysis

(e) Topic 22

Rank	Feature
1	product names technology chemicals
2	brands products
3	brands products
4	products companies
5	companies products services
6	company product names corporate entities
7	brands companies
8	retail brands pricing information
9	companies brands
10	branding product names technology media

(f) Topic 23

Figure 10. Top 10 features (by proximity) - Topics 18–23

Rank	Feature
1	incidents events legal medical
2	events destruction removal
3	incidents discussions financial opportunities risks
4	situations involving emergency responses evacuations
5	incidents safety injuries
6	information disaster declarations implications
7	actions events destruction deletion
8	major incidents aftermath
9	events involving injuries emergencies
10	incidents reports involving emergency situations

(a) Topic 24

Figure 11. Top 10 features (by proximity) - Topic 24

Appendix B. Neural Network Predictions

	Full embedding	Sparse features											
		5	10	30	50	100	300	500	1000	2000	3000	4000	5000
EW Sharpe	4.89 (0.015)	3.00 (0.000)	3.51 (0.000)	4.49 (0.000)	4.92 (0.000)	5.27 (0.092)	5.43 (0.197)	5.44 (0.255)	5.46 (0.159)	5.64 (0.623)	5.55 (0.357)	5.52 (0.279)	5.57
Avg Daily Accuracy	51.08%	50.44%	50.61%	51.08%	51.23%	51.39%	51.51%	51.40%	51.46%	51.60%	51.48%	51.53%	51.60%
Total Accuracy	50.94%	50.27%	50.44%	51.00%	51.13%	51.30%	51.43%	51.29%	51.36%	51.46%	51.35%	51.42%	51.45%

Table VI. Return Predictions and Number of Features
 NN net version of Table [II](#)