Sparse Autoencoders Find Partially Interpretable Features in Italian Small Language Models

Alessandro Bondielli[1,2,*], Lucia Passaro[1,2] and Alessandro Lenci[2]

[1]*Department of Computer Science, University of Pisa*

[2]*CoLing Lab, Department of Philology, Literature and Linguistics, University of Pisa*

Abstract

Sparse Autoencoders (SAEs) have become a popular technique to identify interpretable concepts in Language Models. They have been successfully applied to several models of varying sizes, including both open and commercial ones, and have become one of the main avenues for interpretability research. A number of approaches have been proposed to extract latents from the model, as well as automatically provide natural language explanations for the concepts they supposedly represent. Despite these advances, little attention has been given to applying SAEs to Italian language models. This may be due to several factors: i) the small number of Italian models; ii) the costs associated with leveraging SAEs, which includes the training itself, as well as the necessity to parse and assign an interpretation to a very large number of features.

In this work, we present an initial step toward addressing this gap. We train a SAE on the residual stream of the `Minerva-1B-base-v1.0` model, for which we release the weights; we leverage an automated interpretability pipeline based on LLMs to evaluate both the quality of the latents, and provide explanations for some of them. We show that, albeit the approach shows several limitations, we find some concepts in the weights of the model.

Keywords

Mechanistic Interpretability, Sparse Autoencoders, Large Langauge Models, Italian,

1. Introduction

The rise of Large Language Models (LLMs) have profoundly affected the landscape of Natural Language Processing (NLP). These models have demonstrated remarkable capabilities in many tasks, often achieving near-human performances and saturating benchmarks as soon as they are released. Nevertheless, many questions remain about their internal workings: Whether and how they perform some form of reasoning [1], and to what extent their grasp of concepts through natural language approximates human conceptual understanding.

The aim of **Mechanistic Interpretability** (MechInterp) is to address this pressing issue by attempting to reverse-engineer the learned representations and algorithms within their neural networks [2]. A promising technique within MechInterp is the use of sparse dictionary learning methods like Sparse Autoencoders (SAEs) [3]. The idea behind SAEs is similar to that of standard autoencoder. Autoencoders are unsupervised models that learn two functions: an encoding function, that projects the input data from an n dimensional space into a k != n dimensional space; a decoding function, that should reconstruct the k-dimensional data back into the original n-dimensional one. Autoencoders are typically used for dimensionality reduction, i.e., $k << n$. In the case of SAEs, instead, $k >> n$: the model is trained to project the input space into a much higher-dimensional (and thus sparser) one, and then project it back into the original dimensional space. In our context, SAEs are trained to reconstruct the internal activations of a language model's residual stream by projecting them into a higher-dimensional latent space, while being constrained to use only a small number of "features" from a learned dictionary. This *sparsity* constraint encourages the SAE to learn a set of *monosemantic* features, also referred to as *latents*, that is, features each corresponding to a single, hopefully more interpretable concept [4]. This is in contrast with a *polysemantic* representation, which is typical of standard dense neural networks [5, 6], in which several concepts are superimposed in the same activation patterns. SAEs allow to decompose model activations into a set of near-orthogonal, i.e., largely disentangled features that should be semantically coherent.

Recent work has demonstrated the effectiveness of SAEs in uncovering meaningful features within both toy models [7] and large-scale commercial LMs, revealing representations for concepts ranging from concrete objects to abstract ideas [8, 9, 10]. As noted in [9], several distinctive features have been identified in Claude-3.5-Sonnet – most notably, one corresponding to the "Golden Gate Bridge." SAEs have also been applied successfully to smaller, English-centric models in the 1 to 10 Billion

CLiC-it 2025: Eleventh Italian Conference on Computational Linguistics, September 24 — 26, 2025, Cagliari, Italy

[*]Corresponding author.

[†]These authors contributed equally.

alessandro.bondielli@unipi.it (A. Bondielli); lucia.passaro@unipi.it (L. Passaro); alessandro.lenci@unipi.it (A. Lenci)

0000-0003-3426-6643 (A. Bondielli); 0000-0003-4934-5344 (L. Passaro); 0000-0001-5790-43086 (A. Lenci)

parameter range [11]. This class of models is becoming more and more relevant, as research on Small Language Models (SLMs) [12] and Baby Language Models (BabyLMs) [13, 14], that mitigate the costs of training and serving LLMs while attempting to retain most of their abilities, is a very active endeavour particularly in the open-source/open-weights community.

Two key limitations remain for the applicability of SAEs to achieve interpretability. First, the computational cost of training a SAE. Given their nature, the internal layer of a SAE has to be a number of times larger than the size of residual stream, and thus the context window, of the target LM. The number of parameters of a SAE scales with a factor of the context size of the model, multiplied by the number of *hookpoints* in the models where activations are collected (e.g., after every transformer block/layer). Thus, the larger the target LM, the bigger and the more computationally expensive the SAE.

Second, and most importantly, SAEs output a large number of features, that have then to be interpreted in some way. While the literature has not reached a consensus on what is the best practice, a popular method to address this is to leverage another LLM to provide explanations for the features based on examples of which tokens (and respective contexts) they fired on. For example, if the feature f_i fired on 10 tokens, the explainer model is fed with these tokens, their contexts and the request to find a common property among them. In most works, commercial LLMs with hundred of billions of parameters are successfully used for this task [9, 10]. However, researchers have also shown that smaller and cheaper LMs can be leveraged effectively as well [15].

The vast majority of efforts regarding the use of SAEs for interpretability has been done on English-centric LMs[9, 10, 11]. In addition to this, several efforts have been made in the direction of finding universal features that apply across models and languages [16, 17]. However, models primarily trained on languages other than English have received less attention.

In this work, we aim to provide an early evaluation on the feasibility of using SAEs to interpret models trained to be natively Italian. In the interest of maintaining a limited computational cost, we chose to use the `Minerva-1B-base-v1.0` from the Minerva model family [18]. We trained a SAE on the residual stream of every layer of the model using an Italian split of mC4 [19]. Then, we collected feature activations for the Italian dump of Wikipedia [20], and attempt to explain them and score explanations automatically using an LLM, following [15].

Our contributions are the following:

- We **train and release a Sparse Autoencoder on `Minerva-1B-base-v1.0`**. We make the Autoencoder weights available to the research community via HuggingFace.[1]
- We collect feature activations from a relatively large collection of Italian data, and provide a **quantitative and qualitative evaluation on the explanations using an auto-interpretability pipeline**. We show that SAE are promising for finding concepts in Italian SLMs, but auto-interpretability pipelines shows several limitations for Italian.
- We report on the **challenges and lessons learned on training and using SAEs**, especially in computationally constrained settings.

This paper is organised as follows: In Section 2 we detail the training procedure of the SAE; Section 3 provides an overview of the auto-interpretability pipeline we employ; in Section 4 we present and discuss the obtained results; finally, Section 5 draws some conclusions and highlights future works.

2. SAE Training

In the following, we detail the data and procedure used to train the SAE on the `Minerva-1B-base-v1.0` SLM.

We trained the SAE on the residual stream of the model, with hookpoints on the outputs of each attention block. For our experiments we used the Sparsify library from EleutherAI,[2] which is built to roughly follow the training recipe presented in [10] for a GPT-4 SAE. It trains a k-Sparse Autoencoder [21]. The autoencoder uses a TopK activation function that allows for direct control over the number of active latents. Specifically, it only keeps the k largest latents and assign zero to the rest. Authors in [10] argue that this eliminates the need for the L1 penalty, which biases activations toward zero and is only a rough proxy for L0, and supports any activation function. They also show that it outperforms ReLU autoencoders in sparsity-reconstruction tradeoffs and enhances monosemanticity as small activations are clamped to zero.

Recipe. A full breakdown of the most relevant parameters selected for training is presented in Table 1. The parameters were chosen following recipes for similar sized models, e.g. [11]. The expansion factor controls the size of the hidden layer, and is a multiplier over the model context size. In our case, an expansion factor of 32 yields a hidden layer of size $2,048 \times 32 = 65,536$ parameters.

[1]https://huggingface.co/alessandrobondielli/sae-Minerva-1B-32x The model can be used with the Sparsify and Delphi libraries for interpretabilty.
[2]https://github.com/EleutherAI/sparsify

Parameter	Value
Activation	TopK
Expansion Factor	32
k	32
Multi TopK	False
Transcode	False
Batch Size	16
Loss Function	Fraction of Variance Unexplained (FVU)
Optimizer	Signum

Table 1
Parameters for the SAE training.

Data. As for the training data, we chose to use mC4 [22]. Specifically, we consider the "tiny" split of the `clean_mc4_it` dataset [19]. It includes 6 Billion tokens (4 Billion words). The choice of the dataset was made on the basis that it is relatively large, especially for the Italian language, and it includes a variety of different texts. The data was not included in the training set for `Minerva-1B-base-v1.0`. We chose to use 6 Billion tokens following recent literature on training SAEs for similar-sized models [11].

Setup. We trained our model on a single Nvidia A100 with 80 GB VRAM. A full training run required 200 GPU hours, which roughly equates to 8 days. The final model, that we call **`sae-Minerva-1B-32x`**, occupies around 40 GB of disk space including hookpoints to all layers. The final model is available on HuggingFace[3] and can be loaded and used with Sparsify.

3. Auto-Interpretation of Features

For finding and explaining latents of the SAE models, we use the auto interpretability pipeline proposed in [15]. It is implemented via the Delphi library from EleutherAI.[4] The library includes tools for generating and scoring text explanations for SAE.

The auto intepretability pipeline has three main steps:

1. Activations are collected from a text dataset.
2. An *Explainer* LLM is shown activating contexts and is asked to provide interpretations in natural language for them.
3. A *Scorer* LLM is tasked to distinguish between activating and non activating contexts of a feature, as a binary classifier. This is achieved by asking the model, given several sequences and an intepretation, whether each of the sequences activates the SAE latent with that interpretation.

In the following we detail our implementation of the pipeline.

Collecting Activations. As for the text dataset, we chose to use 20 Million tokens from the Italian subset of the November 2023 Wikipedia dump [20] available on HuggingFace.[5] The choice of Wikipedia as our test dataset rather than a sample of the SAE training data (`clean_mc4_it`) was made with the purpose of increasing the probability of finding concepts specific to the Italian language and culture, that could have been left out from a relatively small sample of a web-based dataset. We created equal-sized batches from the texts, shuffled them, and then collected their token-level activations. We collected the activations at three hookpoints, namely at layers 2, 8 and 14. We did so with the aim of understanding whether there is any difference in the features found near the beginning, middle, or near the end of the residual stream. In the following we use the hookpoint notation to refer to layers, namely `Layer.`x.

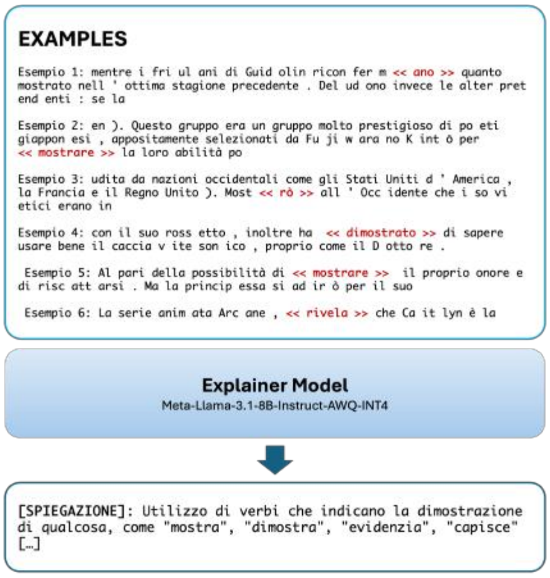
Generating Explanations. As for the explanation generation step, we followed the same procedure as [15]. We showed the Explainer LLM 40 examples of the activating tokens and their contexts. We used a context length of 32 tokens. The activating token can be in any of the 32 positions, but is highlighted as `"« token »"`. We show an example of explanation generation in Figure 1.

To limit the computational cost, we attempted to generate explanations only for a sample of 2,000 latents selected from the pool of 65k. Latents with less than 40 examples were skipped. We used the number of latents with enough examples at each hookpoint in the residual stream to highlight their differences.

The chosen model to generate explanations is `Meta-Llama-3.1-8B-Instruct-AWQ-INT4`,[6] a quantized version of `Meta-Llama-3.1-8B-Instruct` [23]. We prompted the model both in English and Italian. For the English prompt, we used the one provided in [15] for the zero-shot experiment. The Italian version is a direct translation of the English prompt. The translation was made semi-automatically: first, the prompts were translated with Gemini-2.5 Pro.[7] Then, the translated prompt was manually revised to ensure its quality.[8]

Scoring Explanations. Finally, we scored the explanations. We employed a binary classification method. For each explanation, the model was shown five examples of sentences, where each had equal probability of being associated with the latent. The model was then asked

[3]https://huggingface.co/alessandrobondielli/sae-Minerva-1B-32x
[4]https://github.com/EleutherAI/delphi
[5]https://huggingface.co/datasets/wikimedia/wikipedia
[6]https://huggingface.co/hugging-quants/Meta-Llama-3.1-8B-Instruct-AWQ-INT4
[7]https://gemini.google.com, accessed in June 2025.
[8]See Appendix A for the prompts in both languages.



EXAMPLES

```
Esempio 1: mentre i fri ul ani di Guid olin ricon fer m << ano >> quanto
mostrato nell ' ottima stagione precedente . Del ud ono invece le alter pret
end enti : se la

Esempio 2: en ). Questo gruppo era un gruppo molto prestigioso di po eti
giappon esi , appositamente selezionati da Fu ji w ara no K int ò per
<< mostrare >> la loro abilità po

Esempio 3: udita da nazioni occidentali come gli Stati Uniti d ' America ,
la Francia e il Regno Unito ). Most << rò >> all ' Occ idente che i so vi
etici erano in

Esempio 4: con il suo ross etto , inoltre ha << dimostrato >> di sapere
usare bene il caccia v ite son ico , proprio come il D otto re .

Esempio 5: Al pari della possibilità di << mostrare >> il proprio onore e
di risc att arsi . Ma la princip essa si ad ir ò per il suo

Esempio 6: La serie anim ata Arc ane , << rivela >> che Ca it lyn è la
```

```
[SPIEGAZIONE]: Utilizzo di verbi che indicano la dimostrazione
di qualcosa, come "mostra", "dimostra", "evidenzia", "capisce"
[…]
```

Figure 1: Explanation Generation with examples. Activating tokens are marked as « `token` ». In the Figure we highlight them also in **bold red**.

to decide, for each example, whether it corresponded to the explanation, and output a list of of decisions. If the output did not match a list of decision, it was assigned `None`. The output was then compared with the ground truth provided by the activations. The model for scoring was the same one used to generate explanations. As for the prompt and its translation in Italian, we followed the same translation procedure as well. We evaluated the quality of explanations with accuracy. Specifically, we considered a **per-sample accuracy** (i.e., how many out of the five examples the scorer model got right) and the average accuracy across across latents for the same hookpoint.

We acknowledge that our choice of using a multilingual, relatively small, and quantized LLMs for generating and scoring explanations is far from ideal, and it is not an adequate substitute neither for human evaluation nor for more performing LLMs. The choice of a multilingual model rather than an Italian-only one was made due to the current lack of such models with open weights, high performances and capability to follow instructions. This choice led also to prompting the model both in English and Italian; this was done to assess its explanation/scoring capabilities both in its "native" language, albeit on data from another language, and on Italian, in order to limit potential biases in the interpretation of results from using only one or the other language. As for the choice of a medium-sized quantized model, this was made in the interest of limiting the computational costs of our experiments, i.e., both in terms of the memory footprint of the model, and of the overall GPU hours. Using larger (including non-quantized variants) models would have drastically increased both the need of resources and overall time of the experiments. Nonetheless, we argue that our choice represents a lower-cost alternative to using much larger and costlier models, that could prove especially useful to provide some early insights into the quality of the latents found by the SAE, and of the model being interpreted.

Authors in [15] estimate a cost in the order of hundreds or thousand of dollars for explaining and scoring 100k latents with larger or commercial models; our experiments, in contrast, can be easily replicated on a single GPU. In our case, generating and scoring explanations for 2,000 latents at three different hookpoints, in two different languages, took 0.5 GPU hours each on a single Nvidia A100, for a grand total of 3 GPU hours. Given the size of the model used, the experiments could be also replicated on much less performing hardware as well, provided a trade-off on GPU hours.

4. Results and Discussion

In the following, we present our results. First, we show a quantitative evaluation of the extracted latents, and the performances of the generation and scoring pipeline, both with Italian and English prompts. To explore the results in greater depth, we also perform a qualitative evaluation. We consider explanations that received highest scores by the scorer model. We use the results to discuss the feasibility of the proposed approach on Italian SLM, as well as potential shortcomings.

4.1. Quantitative Evaluation

The core of our quantitative analysis is based on the results we obtained using the Delphi library, with the configuration presented in Section 3.

Quality of the Latents. To evaluate the quality of the latents obtained via the SAE encoding, several metrics can be used. Recall that we collected latent activations using 20 Million tokens from the Italian subset of Wikipedia. Note also that here we are not yet using prompts, so we do not distinguish between Italian and English.

Table 2 provide several common metrics used to evaluate the quality of the extracted latents at each hookpoint. First, we look at fraction of alive latents. A latent is considered alive if at least one input token in the dataset made it fire. With the exception of Layer.8, the other two have much smaller fractions of alive latents than it is typical for SAEs (see for examples results reported in

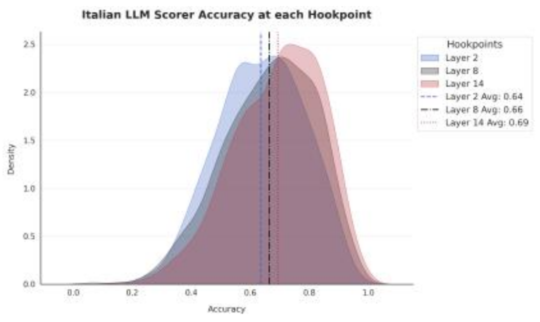


Figure 2: Accuracy distribution with Italian prompts.

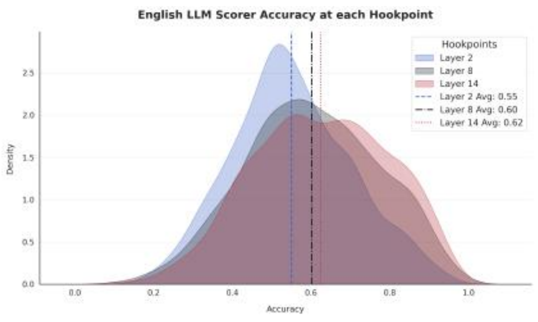


Figure 3: Accuracy distribution with English prompts.

Metric	Layer.2	Layer.8	Layer.14
Fraction of latents alive (%)	72.02	**95.16**	84.65
Latents fired $>1\%$ of the time (%)	0.27	**0.45**	0.38
Latents fired $>10\%$ of the time (%)	**0.06**	0.00	0.01
Weak single-token latents (%)	**9.93**	2.20	2.77
Strong single-token latents (%)	**12.40**	0.55	0.47

Table 2
Latent activity statistics across selected layers

[10] and [11]). This may be the results of several factors. On the SAE side, we could hypothesize an overcomplete latent space for the evaluation data, i.e. a too broad latent space for encoding the evaluation data. Recall in fact that we used mC4 to train the SAE, and evaluated it on Wikipedia, which may present less variety in terms of texts.

On the Language Model side, we could hypothesize that the latent space of the analyzed model is very anisotropic at both earliest and latest layers, while more isotropic near the middle of the stack. This however is in direct contrast with works such as [24], and thus requires a more in-depth analysis, that we leave to future works. Another interesting aspect to consider are weak and strong single-token latents, that is latents that fire on a specific token only. Weak ones are those for which the token in question makes many other latents fire; strong ones are cases where the token preferentially activates the specific latent. We observe that `Layer.2` is heavily biased towards single token latents. This may indicate that earliest layers sill leverage the embedding representation quite strongly. Finally, we see that latents that fired either more than one or 10% of the times are less and less as we move towards the residual stream. These latents may be used to store single-token concepts of words such as function ones.

Quality of the Explanations. To evaluate the quality of explanations, we consider the results of the explanation generation and scoring pipeline. Specifically, for each latent, we compute the accuracy at distinguishing between sequences that activate and do not activate the latent. Figures 2 and 3 show respectively the distribution of Accuracy for the scorer model using Italian and English prompts for each hookpoint in the residual stream.

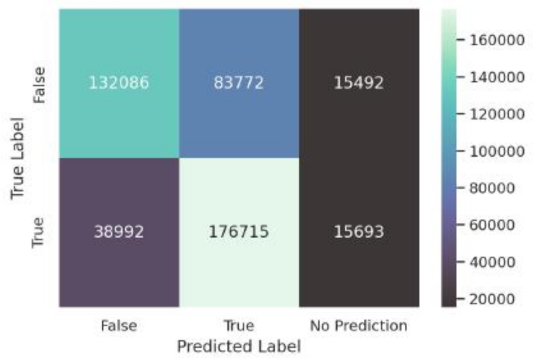
We observe that, in both cases, there are significant differences both in distribution and averages for the three hookpoints. We also observe that explanations for latents extracted from later layers seem to be easier to score correctly for the scorer model. This may indicate that **concepts identified in later layers are, on average, more easily interpretable by an LLM**. The accuracy scores obtained using the Italian prompt are generally higher than those for the English one, with average scores ranging from 0.64 to 0.69; the English ones, in contrast, range from 0.55 to 0.62. However, these results in isolation cannot be taken as a direct indication that explanations in Italian are better than English ones. It may as well be the result of poorer and broader explanations provided by the Explainer model.

We also plot the aggregate confusion matrices over all the predictions of both prompts. The confusion matrices are shown in Figure 4. While the model prompted in Italian seem to fare better in all metrics except for True Positives, we also see that the number of times the model was not able to follow instructions and provide a prediction with the Italian prompt is three times higher than with the English one. This may be further indication that the Explainer/Scorer model used struggles with Italian.

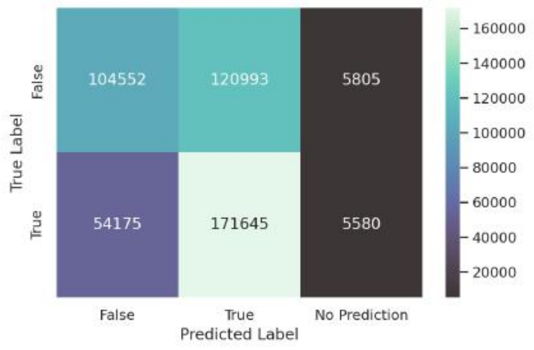
4.2. Qualitative Evaluation

To dig deeper into the quality of the explanations, we directly looked at them and provide examples of seemingly good and bad explanations. Specifically, we sampleed from the 50 explanations that received highest scores by the Scorer, both in English and Italian.

As for the Italian explanations, we immediately observed that a large fractions of them suffer from *Degenerate Repetition* [25]: The model starts to generate the same token or sequence of tokens over and over. On



(a) Italian Prompt.



(b) English Prompt.

Figure 4: Confusion Matrices for the Scorer model on both the Italian and English prompt.

the contrary, English ones does not suffer from this issue. However, if we look at the quality of explanations, aside from repetitions, we observe that at least some of the Italian ones are quite relevant to the examples, and while sometimes slightly missing the mark, they highlight some interesting aspects of the tokens that fire the latent.

Among these, we can clearly see that `Layer.2` is mostly represented by single token latents: the token "ale" as part of "*federale*" (federal), in several contexts, or the token "*letto*", as both a noun (bed) and a verb (read). `Layer.14` latents on the other had appear to represent more abstract concepts. For example, we see latents firing on the final number of a year date, and a very interesting latent firing on the concept of *competition* (see Fig. **??**). `Layer.8` explanations are generally more confusing and less interesting. Examples are reported in Figure 5 with the relative explanation, cut to avoid showing repetitions.

As for the English explanations on the other hand, we observed that most of them actually miss the mark. In fact, they often provide an explanation related to the contexts, rather than the firing tokens. This may be due to the fact that, while it is specified in the prompt, we use Italian texts as examples but instructions and expected outputs are in English. Neverhteless, we observe an interesting trend: most explanations, at all layers, that actually focus on the firing tokens refer to functional aspects of the text, including punctuation marks, special characters, and functional words. For example, Latent 1818 of `Layer.14` is explained as "Prepositions and conjunctions used to connect words or phrases in Italian text, such as "*a*", "*di*", "*nel*", "*in*", "*su*", "*da*", "*al*", "*nei*", "*all*", "*sulle*", "*col*" [...]". This is in contrast with what we observed for Italian explanations.

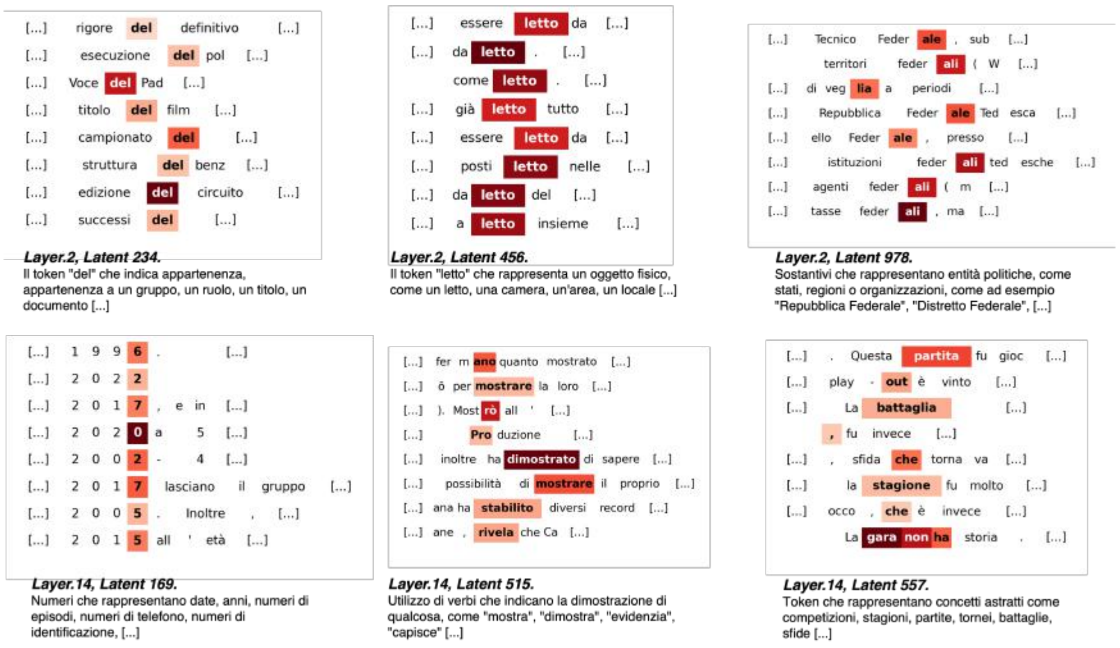
4.3. Discussion of Key Findings

In the following, we highlight some of the key aspects that emerged from the experiments.

SAEs can find partially interpretable features in Italian Small Language Models. First, we observe that using a SAE we are able to extract features that somewhat align to interpretable concepts, despite some limitations that we can mostly attribute to the quality of the training data, both for the original model and the SAE, and to the limitations of the auto-interpretability pipeline (see below). It is possible that leveraging a dataset more attuned with the Italian culture would yield better results in finding relevant latents.

Different behaviours in the residual stream. We observed some relevant differences in the quality and types of latents that are properly identified in various points of the residual stream. In general, we observed that latents obtained from earlier in the stream are more relevant to single tokens and grammatical aspects of the language, while latents in later points of the stream show a slight tendency towards more abstract conceptualizations.

Auto-interpretability is promising, but currently shows limitations for Italian. Auto-interpretability pipelines are definitely a promising approach for simplifying and reducing the costs of finding explanations for latents of SAEs. Our experiment suggest in fact that this is a low-cost alternative that is nonetheless able to deliver some interesting results. Nevertheless, we observed two main limitations that we can argue are actually two sides of the same coin. On the one side, the Explainer model showed some limitations in understanding the task and providing coherent texts for the explanations, while the Scorer model performed quite poorly in the binary classificationt task. This is especially true in the case of language mixing, i.e. when the model is prompted in its "main" language, i.e. English, but has to work on another


Layer.2, Latent 234.
Il token "del" che indica appartenenza, appartenenza a un gruppo, un ruolo, un titolo, un documento [...]

Layer.2, Latent 456.
Il token "letto" che rappresenta un oggetto fisico, come un letto, una camera, un'area, un locale [...]

Layer.2, Latent 978.
Sostantivi che rappresentano entità politiche, come stati, regioni o organizzazioni, come ad esempio "Repubblica Federale", "Distretto Federale", [...]

Layer.14, Latent 169.
Numeri che rappresentano date, anni, numeri di episodi, numeri di telefono, numeri di identificazione, [...]

Layer.14, Latent 515.
Utilizzo di verbi che indicano la dimostrazione di qualcosa, come "mostra", "dimostra", "evidenzia", "capisce" [...]

Layer.14, Latent 557.
Token che rappresentano concetti astratti come competizioni, stagioni, partite, tornei, battaglie, sfide [...]

Figure 5: Examples of explanations for latents in Italian.

language, in this case Italian. On the other side, the size of the model used in our experiments could severely limit its performances.

Thus, both issues could be solved either by leveraging a stronger Italian-centric model as the Explainer/Score, or by using a generally larger and better performing model. However, as for the first solution, there are currently no models on par with English ones in the 7-15B parameters range, wich whould allow for reducing the cost. As for the second solution, this would dramatically increase the costs, both computational and monetary.

5. Conclusions and Future Works

In this paper, we have shown that SAEs can partly uncover interpretable concepts in Italian Small Language Models. Specifically, we did so by training a SAE model on the residual stream of the `Minerva-1B-base-v1.0` SLM, and then applying an auto-interpretability pipeline to generate explanations for its latents.

Our findings suggest that SAE can be used to this end, and that it exist a hierarchical representation within the model, with earlier layers showing more token-centric features and later layers more abstract concepts. As for the auto-interpretability pipeline, while promising for its low cost, underscored the need for better language-specific tools for Italian.

Moving forward, we aim to explore several avenues. First, we plan to scale our experiments in two directions: on the one hand, we aim to train SAEs on larger Italian models, e.g. larger variants of Minerva as well as others; on the other hand, we observe that we need to improve the models used for auto-interpretability, in order obtain more reliable explanations. This could be achieved both by scaling them up substantially, and by tuning Italian-speaking models to the specific tasks of latent explanation and scoring. Second, we plan to leverage SAE and auto interpretability to address potential differences of representations in models pre-trained specifically on Italian data, e.g. Minerva and Velvet [26], and multilingual models that received only fine-tuning in Italian, like the LLaMAntino variants [27] and Cerbero [28]. Finally, we plan to explore the larger latent space to attempt to uncover features linked specifically to Italian-centric concepts, in addition to properties of the Italian Language.

This work is an early first step in exploring interpretability research using Sparse Autoencoders for non-English-centric Language Models. Albeit limited in scope, we are optimistic that it may provide a relevant foundation for this yet under explored research area, both in terms of approach and the release of open models for the community.

Limitations

Our initial effort to interpret Italian SLMs using Sparse Autoencoders has several limitations. The choice of the smaller `Minerva-1B-base-v1.0` model, driven by computational constraints, means our findings might not generalize to larger Italian models. The SAE's training data, while substantial for Italian, might not fully capture all linguistic nuances, potentially affecting the quality of learned features. Additionally, using different data to train and evaluate the SAE, while arguably not problematic in principle, may have introduced some unwanted biases.

A key limitation stems from our cost-effective auto-interpretability pipeline, which relies on a relatively small, quantized multilingual LLM. This model struggled with generating coherent Italian explanations, often repeating itself, and performed poorly in scoring when mixing languages. This highlights the strong dependence of explanation quality on the explainer/scorer model's capabilities, and the current lack of robust, affordable, Italian-specific tools.

Finally, our analysis was based on a sample of 2000 latents across only three layers, not the entire SAE latent space. While insightful, this limited scope and subjective qualitative assessment means we cannot yet claim a comprehensive understanding of the model's internal workings.

Acknowledgments

This work has been supported by the PNRR MUR project PE0000013-FAIR (Spoke 1), funded by the European Commission under the NextGeneration EU programme, and the EU EIC project EMERGE (Grant No. 101070918).

References

[1] P. Shojaee, I. Mirzadeh, K. Alizadeh, M. Horton, S. Bengio, M. Farajtabar, The illusion of thinking: Understanding the strengths and limitations of reasoning models via the lens of problem complexity, 2025. URL: https://ml-site.cdn-apple.com/papers/the-illusion-of-thinking.pdf.

[2] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, S. Carter, Zoom in: An introduction to circuits, Distill 5 (2020) e24.

[3] B. A. Olshausen, D. J. Field, Sparse coding with an overcomplete basis set: A strategy employed by v1?, Vision research 37 (1997) 3311–3325.

[4] H. Cunningham, A. Ewart, L. Riggs, R. Huben, L. Sharkey, Sparse autoencoders find highly interpretable features in language models, 2023. URL: https://arxiv.org/abs/2309.08600. `arXiv:2309.08600`.

[5] N. Elhage, T. Hume, C. Olsson, N. Schiefer, T. Henighan, S. Kravec, Z. Hatfield-Dodds, R. Lasenby, D. Drain, C. Chen, R. Grosse, S. McCandlish, J. Kaplan, D. Amodei, M. Wattenberg, C. Olah, Toy models of superposition, Transformer Circuits Thread (2022).

[6] A. Scherlis, K. Sachan, A. S. Jermyn, J. Benton, B. Shlegeris, Polysemanticity and capacity in neural networks, 2025. URL: https://arxiv.org/abs/2210.01892. `arXiv:2210.01892`.

[7] E. Anders, C. Neo, J. Hoelscher-Obermaier, J. N. Howard, Sparse autoencoders find composed features in small toy models, https://shorturl.at/YOtYR, 2024.

[8] T. Bricken, A. Templeton, J. Batson, B. Chen, A. Jermyn, T. Conerly, N. Turner, C. Anil, C. Denison, A. Askell, R. Lasenby, Y. Wu, S. Kravec, N. Schiefer, T. Maxwell, N. Joseph, Z. Hatfield-Dodds, A. Tamkin, K. Nguyen, B. McLean, J. E. Burke, T. Hume, S. Carter, T. Henighan, C. Olah, Towards monosemanticity: Decomposing language models with dictionary learning, Transformer Circuits Thread (2023). https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[9] A. Templeton, T. Conerly, J. Marcus, J. Lindsey, T. Bricken, B. Chen, A. Pearce, C. Citro, E. Ameisen, A. Jones, H. Cunningham, N. L. Turner, C. McDougall, M. MacDiarmid, C. D. Freeman, T. R. Sumers, E. Rees, J. Batson, A. Jermyn, S. Carter, C. Olah, T. Henighan, Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet, Transformer Circuits Thread (2024). URL: https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

[10] L. Gao, T. D. la Tour, H. Tillman, G. Goh, R. Troll, A. Radford, I. Sutskever, J. Leike, J. Wu, Scaling and evaluating sparse autoencoders, arXiv preprint arXiv:2406.04093 (2024).

[11] T. Lieberum, S. Rajamanoharan, A. Conmy, L. Smith, N. Sonnerat, V. Varma, J. Kramar, A. Dragan, R. Shah, N. Nanda, Gemma scope: Open sparse autoencoders everywhere all at once on gemma 2, in: Y. Belinkov, N. Kim, J. Jumelet, H. Mohebbi, A. Mueller, H. Chen (Eds.), Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP, Association for Computational Linguistics, Miami, Florida, US, 2024, pp. 278–300. URL: https://aclanthology.org/2024.blackboxnlp-1.19/. doi:`10.18653/v1/2024.blackboxnlp-1.19`.

[12] B. Yuan, C. Li, C. Zhang, X. Chen, Y. Liu, Y. Zhang, Y. Wu, Z. Wang, Y. Wang, Y. Cao, et al., Small language models: A survey of the state of the art,

arXiv preprint arXiv:2407.01513 (2024).

[13] M. Y. Hu, A. Mueller, C. Ross, A. Williams, T. Linzen, C. Zhuang, R. Cotterell, L. Choshen, A. Warstadt, E. G. Wilcox, Findings of the second BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora, in: M. Y. Hu, A. Mueller, C. Ross, A. Williams, T. Linzen, C. Zhuang, L. Choshen, R. Cotterell, A. Warstadt, E. G. Wilcox (Eds.), The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning, Association for Computational Linguistics, Miami, FL, USA, 2024, pp. 1–21. URL: https://aclanthology.org/2024.conll-babylm.1/.

[14] L. Capone, A. Bondielli, A. Lenci, ConcreteGPT: A baby GPT-2 based on lexical concreteness and curriculum learning, in: M. Y. Hu, A. Mueller, C. Ross, A. Williams, T. Linzen, C. Zhuang, L. Choshen, R. Cotterell, A. Warstadt, E. G. Wilcox (Eds.), The 2nd BabyLM Challenge at the 28th Conference on Computational Natural Language Learning, Association for Computational Linguistics, Miami, FL, USA, 2024, pp. 189–196. URL: https://aclanthology.org/2024.conll-babylm.16/.

[15] G. Paulo, A. Mallen, C. Juang, N. Belrose, Automatically interpreting millions of features in large language models, arXiv preprint arXiv:2410.13928 (2024).

[16] M. Lan, P. Torr, A. Meek, A. Khakzar, D. Krueger, F. Barez, Sparse autoencoders reveal universal feature spaces across large language models, arXiv preprint arXiv:2410.06981 (2024).

[17] J. Lindsey, W. Gurnee, E. Ameisen, B. Chen, A. Pearce, N. L. Turner, C. Citro, D. Abrahams, S. Carter, B. Hosmer, J. Marcus, M. Sklar, A. Templeton, T. Bricken, C. McDougall, H. Cunningham, T. Henighan, A. Jermyn, A. Jones, A. Persic, Z. Qi, T. B. Thompson, S. Zimmerman, K. Rivoire, T. Conerly, C. Olah, J. Batson, On the biology of a large language model, Transformer Circuits Thread (2025). URL: https://transformer-circuits.pub/2025/attribution-graphs/biology.html.

[18] R. Orlando, L. Moroni, P.-L. Huguet Cabot, S. Conia, E. Barba, S. Orlandini, G. Fiameni, R. Navigli, Minerva LLMs: The first family of large language models trained from scratch on Italian data, in: F. Dell'Orletta, A. Lenci, S. Montemagni, R. Sprugnoli (Eds.), Proceedings of the 10th Italian Conference on Computational Linguistics (CLiC-it 2024), CEUR Workshop Proceedings, Pisa, Italy, 2024, pp. 707–719. URL: https://aclanthology.org/2024.clicit-1.77/.

[19] G. Sarti, M. Nissim, IT5: Text-to-text pretraining for Italian language understanding and generation, in: N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, N. Xue (Eds.), Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), ELRA and ICCL, Torino, Italy, 2024, pp. 9422–9433. URL: https://aclanthology.org/2024.lrec-main.823.

[20] W. Foundation, Wikimedia downloads, ???? URL: https://dumps.wikimedia.org.

[21] A. Makhzani, B. Frey, K-sparse autoencoders, arXiv preprint arXiv:1312.5663 (2013).

[22] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, C. Raffel, mT5: A massively multilingual pre-trained text-to-text transformer, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 483–498. URL: https://aclanthology.org/2021.naacl-main.41. doi:10.18653/v1/2021.naacl-main.41.

[23] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al., The llama 3 herd of models, arXiv preprint arXiv:2407.21783 (2024).

[24] A. Razzhigaev, M. Mikhalchuk, E. Goncharova, I. Oseledets, D. Dimitrov, A. Kuznetsov, The shape of learning: Anisotropy and intrinsic dimensions in transformer-based models, in: Y. Graham, M. Purver (Eds.), Findings of the Association for Computational Linguistics: EACL 2024, Association for Computational Linguistics, St. Julian's, Malta, 2024, pp. 868–874. URL: https://aclanthology.org/2024.findings-eacl.58/.

[25] H. Li, T. Lan, Z. Fu, D. Cai, L. Liu, N. Collier, T. Watanabe, Y. Su, Repetition in repetition out: towards understanding neural text degeneration from the data perspective, in: Proceedings of the 37th International Conference on Neural Information Processing Systems, 2023, pp. 72888–72903.

[26] A. Team, Almawave presents velvet: The sustainable and high-performance italian ai, 2025. URL: https://www.almawave.com.

[27] P. Basile, E. Musacchio, M. Polignano, L. Siciliani, G. Fiameni, G. Semeraro, Llamantino: Llama 2 models for effective text generation in italian language, 2023. arXiv:2312.09993.

[28] F. A. Galatolo, M. G. Cimino, Cerbero-7b: A leap forward in language-specific llms through enhanced chat corpus generation and evaluation, arXiv preprint arXiv:2311.15698 (2023).

Explainer Prompt (Eng)

```
You are a meticulous AI researcher
     conducting an important
     investigation into patterns found
      in the Italian language. Your
     task is to analyze text and
     provide an explanation that
     thoroughly encapsulates possible
     patterns found in it.

Guidelines:
You will be given a list of text
     examples in Italian on which
     special words are selected and
     between delimiters like <<this>>.
      If a sequence of consecutive
     tokens all are important, the
     entire sequence of tokens will be
      contained between delimiters <<
     just like this>>. How important
     each token is for the behavior is
      listed after each example in
     parentheses.

- Try to produce a concise final
     description. Simply describe the
     text latents that are common in
     the examples, and what patterns
     you found.
- If the examples are uninformative,
     you don't need to mention them.
     Don't focus on giving examples of
      important tokens, but try to
     summarize the patterns found in
     the examples.
- Do not mention the marker tokens
     (<< >>) in your explanation.
- Do not make lists of possible
     explanations. Keep your
     explanations short and concise.
- The last line of your response must
      be the formatted explanation,
     using [EXPLANATION]:

{{ prompt }}
```

Explainer Prompt (Ita)

```
Sei un meticoloso ricercatore di
     intelligenza artificiale che
     conduce un'importante indagine
     sugli schemi presenti nella
     lingua italiana. Il tuo compito e
     ' analizzare il testo e fornire
     una spiegazione che racchiuda in
     modo esauriente i possibili
     schemi in esso riscontrati.

Linee guida:
Ti verra' fornito un elenco di esempi
      di testo in italiano in cui
     parole speciali sono selezionate
     e inserite tra delimitatori come
     <<questo>>. Se una sequenza di
     token consecutivi e' tutta
     importante, l'intera sequenza di
     token sara' contenuta tra
     delimitatori <<proprio come
     questo>>. L'importanza di ciascun
      token per il comportamento e'
     elencata dopo ogni esempio tra
     parentesi.

- Cerca di produrre una descrizione
     finale concisa. Descrivi
     semplicemente gli elementi
     latenti del testo comuni negli
     esempi e gli schemi che hai
     trovato.
- Se gli esempi non sono informativi,
      non e' necessario menzionarli.
     Non concentrarti sul fornire
     esempi di token importanti, ma
     cerca di riassumere gli schemi
     trovati negli esempi.
- Non menzionare i token marcatori
     (<< >>) nella tua spiegazione.
- Non creare elenchi di possibili
     spiegazioni. Mantieni le tue
     spiegazioni brevi e concise.
- L'ultima riga della tua risposta
     deve essere la spiegazione
     formattata, usando [SPIEGAZIONE]:

{{ prompt }}
```

Figure 6: Explainer prompts in English (original, from [15]), and Italian (translated).

A. Explainer Prompts

In Figure 6 we provide prompts fed to the Explainer model, both in English (original from [15]) and Italian (translation).