INTERPRETO: An Explainability Library for Transformers

Antonin Poché[1,2,*], Thomas Mullor[1], Gabriele Sarti[3], Frédéric Boisnard[4], Corentin Friedrich[1], Charlotte Claye[5,6], François Hoofd[1,7], Raphael Bernas[5], Nicholas Asher[2,8] Céline Hudelot[5], Fanny Jourdan[1,*],

[1]IRT Saint Exupéry Toulouse, [2]IRIT Toulouse, [3]Khoury College of Computer Sciences, [4]Ampere, [5]MICS, CentraleSupélec,[6]Scienta Lab, [7]Thales Avionics, [8]ANITI
[*]**Equal contribution**

Correspondence: antonin.poche | fanny.jourdan @irt-saintexupery.com

Abstract

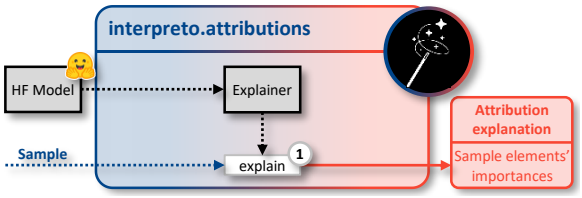
INTERPRETO is an open-source Python library for interpreting **HuggingFace language models**, from early BERT variants to LLMs. It provides two complementary families of methods: **attribution** methods and **concept-based** explanations. The library bridges recent research and practical tooling by exposing explanation workflows through a unified API for both **classification** and text **generation**. A key differentiator is its end-to-end concept-based pipeline (from activation extraction to concept learning, interpretation, and scoring), which goes beyond feature-level attributions and is uncommon in existing libraries. See GitHub or demo website.

1 Introduction

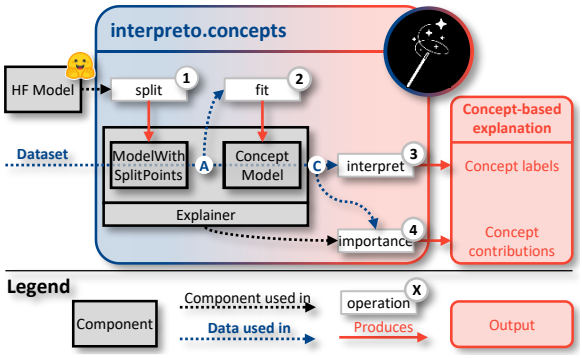
Transformer-based language models are widely deployed for classification and text generation, yet understanding their behavior remains important for debugging, bias detection, or safety. Hence, practitioners need tools to simplify access to explanations, including **attributions** (importance scores assigned to input elements such as tokens or spans) and **concept-based** explanations (based on higher-level features learned from activations).

Numerous libraries support attributions or concept explanations (Tabs. 1 and 2). However, these capabilities are often split across packages or tailored to specific modalities or tasks, increasing pipeline complexity. Moreover, documentation or metrics are sometimes missing.

We present INTERPRETO, an open-source library for interpreting HuggingFace NLP models using both attribution methods and concept-based activation-level analysis (a component of mechanistic interpretability) (Fig. 1). INTERPRETO provides a unified API for classification and text generation, along with visualization tools, metrics, tutorials, and documentation. Our goal is to make research methods easier to adopt and to support benchmarking. We welcome issues and pull requests.



(a) `attributions` module (Sec. 5): a single call explains a sample prediction via element-wise importance scores.



(b) `concepts` module (Sec. 6): **(1) Split** a HF model and run the dataset to collect latent activations A; **(2) Fit** a concept model on A and encode activations into concept activations C; **(3) Interpret** concepts by linking the dataset with C to produce human-readable labels; **(4) Score** concept contributions by computing importance of C w.r.t the model output.

Figure 1: INTERPRETO pipelines for HuggingFace (HF) language models. The same API applies to both **classification** and **generation** models.

Availability. INTERPRETO is open-sourced on GitHub[1] under the **MIT license**. The library comes with its documentation[2] and tutorials[3]. The demonstration website[4] and video[5] can be accessed freely.

How to run. INTERPRETO should be installed via `uv pip install interpreto`. Examples from Figs. 2, 3, 5, and 8a are runnable end-to-end with only INTERPRETO; a few optionally use `datasets` or an external LLM API for labeling.

[1] https://github.com/FOR-sight-ai/interpreto
[2] https://for-sight-ai.github.io/interpreto/
[3] https://for-sight-ai.github.io/interpreto/notebooks/attribution_walkthrough/
[4] https://for-sight-ai.github.io/interpreto-demo/
[5] https://youtu.be/CXep27D3abM

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 1–13

	Splitting	Learning	Interpretation	Contributions	Metrics	pip & doc
INTERPRETO	✔	✔	✔	✔	✔	✔
NNsight / NNterp / TransformerLens	✔	✘	✘	✘	✘	✔
Overcomplete	●	✔	●	✔	✔	✔
SAELens	✘	✔*	✘	●	✔	✔
Sparsify	✘	✔*	✘	●	✔	●
ViT-Prisma	●	✔*	●	✘	✔	✘
Neuronpedia	✔	✘	✔	✔	✔	●
Delphi	✘	✘	✔	✘	✔	✘
SAE-Vis	✘	✘	✔	✘	✘	✘
SAEBench	✘	✘	✔	✘	✔	●

Table 1: Comparison of libraries for post-hoc unsupervised concept-based. Which steps from Sec. 6 are available? (✔ supported; ✘ not supported; ● either not for language models, not directly, or not both; * only SAEs). "pip & doc" if existing pip package and documentation. We grouped libraries in four categories: 1) Model splitting (model agnostic access to activations); 2) Concepts learning; 3) Interpretations and visualizations; and 4) Benchmarks.

	Cls	Gen	Metrics	Simple viz	Gran
INTERPRETO	✔	✔	✔	✔	✔
Captum	✔	✔	✔	✘	✘
Ferret	✔	✘	✔	✘	✘
Inseq	✘	✔	✘	✘	✘
SHAP	✔	✔	✘	✔	✘

Table 2: Comparison of language-attribution libraries. (✔ supported; ✘ not supported). **Cls**: sequence classification; **Gen**: text generation; **Metrics** refer to faithfulness metrics; **Simple viz**: interactive generation visualization; **Gran**: granularity (token, words, sentences).

Contributions.

- **`attribution` module (Sec. 5)** supports HuggingFace models for classification and text generation, with 10 attribution methods, 2 evaluation metrics, and attributions' granularity control.
- **`concepts` module (Sec. 6)** provides an end-to-end concept-based pipeline. It wraps `nnsight` (Fiotto-Kaufman et al., 2024) to split HuggingFace language models. It offers 15 concept-learning options (mostly via `overcomplete` (Fel, 2025)), 3 concept-interpretation methods, concept-importance estimation, and 7 metrics.

2 Related work

For language models' interpretability, open-source libraries fall into two categories: attribution and mechanistic interpretability (MI) libraries. We further detail related libraries in appendix A.

Tab. 2 compares INTERPRETO with attribution libraries, including SHAP (Lundberg and Lee, 2017) and Captum (Kokhlikyan et al., 2020). INTERPRETO differs by supporting explanations for both classification and text generation, providing evaluation metrics, and exposing granularity controls.

Tab. 1 compares INTERPRETO with MI libraries. These can be grouped into four categories: 1) tooling for model splitting, such as NNsight (Fiotto-Kaufman et al., 2024); 2) concepts learning, such as SAELens (Bloom and Chanin, 2024); 3) concepts interpretations, such as Delphi (Paulo et al., 2025); and 4) benchmarks, such as SAEBench (Karvonen et al., 2025). INTERPRETO's main differentiator is that it integrates these steps into a single package with documentation and examples, reducing the effort needed to apply them in practice.

3 System overview

Environment. INTERPRETO is available and was tested with python from 3.10 to 3.13, torch$>=$ 2.0, transformers$>=$ 4.22, and nnsight$>=$ 0.5.1.

Supported models and tasks. INTERPRETO supports classic encoders, decoders, and encoder-decoders via the HuggingFace API (tested list in appendix B). Concretely, the `attributions` module supports `SequenceClassification` and `CausalLM` models. The `concepts` module also supports `MaskedLM` and `Seq2SeqLM`.

Workflow and Interface. INTERPRETO has two modules; their main components are shown in Fig. 1 and described in Secs 5 and 6. In the `attributions` module, an explainer takes a model and an input sample and produces token- or word-level importance scores for the model prediction. In the `concepts` module, the workflow is: i) split model and extract an activation dataset; ii) learn concepts as recurring patterns in these activations; iii) interpret the learned concepts; and iv) estimate their importance for predictions.

Evaluation. To validate correctness, we provide unit tests for all functions and test more than 15 model architectures (Appendix B). We also include sanity checks on manually constructed models in both tasks to verify the correctness of explanations. Finally, Mussot et al. (2026) used INTERPRETO in an industrial setting, supporting its accessibility and practical usefulness.

4 Demonstration

Content. The demonstration material is a website that serves as a gallery of INTERPRETO explanations. Appendix C is a screenshot of the website. Each explanation includes a button that opens a minimal, runnable snippet.

Requirements. The demonstration website only requires an internet connection, since explanations are precomputed. The snippets are end-to-end runnable with INTERPRETO alone, with optional dependencies for data loading (`datasets`) and concept labeling (external LLM API).

Types of explanations. The available explanations, interactions, and visuals depend on the task, the explanation family, and user-specified settings. The gallery covers: i) **Attributions for classification**: examples that explain all classes in parallel or only the predicted class; ii)**Attributions for generation**: users select an output element and view the corresponding attributions; iii) **Concepts for classification**: global concept-based explanations, either general across classes or learned class-wise; and iv) **Concepts for generation**: local concept-based analyses for individual generations.

Website interface. All explanations are interactive: depending on the view, users can select classes, output tokens, or concepts. Users can choose the task (classification/generation), model, dataset, explanation family (attribution/concepts), method subset, and the instance to inspect.

Models and datasets. All models are hosted on HuggingFace; generating explanations requires a GPU (see computation times). The gallery covers 6 models: 3 classifiers (DistilBERT/IMDB (Sanh et al., 2019; Maas et al., 2011), BERT/emotion (Devlin et al., 2019; Saravia et al., 2018), RoBERTa/AG-News (Liu et al., 2019; Zhang et al., 2015)) and 3 generators (GPT-2 (Radford et al., 2019), Qwen3-0.6B (Yang et al., 2025), Llama 3.1 8B (Grattafiori et al., 2024)).

User scenario. A practitioner debugs an emotion recognition model. They first explore the explanations on the demo website. From which, they copy a runnable code snippet for KernelSHAP attributions and adapt it for their use case. Although highlighted tokens appear plausible, predictions remain incorrect in several cases. They then switch to class-wise concept-based explanations in the gallery and reproduce the concept discovery step locally (e.g., Semi-NMF). Inspecting the global concepts' importance suggests that "fear" and "sadness" are not well separated; further checks reveal an inconsistent class-index mapping across data sources, explaining the observed errors.

Computation times. Website explanations are precomputed. Attributions typically require 10–100 forward passes or 5–20 gradient computations (seconds). Concept pipelines are dominated by activation extraction and concept importance scoring; small runs take minutes on an RTX 3080 (Fig. 5), while large SAEs can take hours.

5 `interpreto.attributions`

Definition and taxonomy. Attribution explains a prediction by estimating the contribution of input features. In vision, this yields heatmaps; in NLP, it highlights salient tokens or words (Figs. 2 and 3). Methods fall into two broad families. **Perturbation-based** approaches modify the input and measure the effect on the output. **Gradient-based** approaches use derivatives of the output with respect to the input. These families are closely related: gradients correspond to the limit of infinitesimal perturbations.

API. With INTERPRETO (as in other attribution libraries listed in appendix A) the workflow has three steps (summarized in Fig. 1a and illustrated with code examples in Figs. 2 and 3):

1. Instantiate an `AttributionExplainer` with a HuggingFace model and tokenizer.
2. Compute explanations for `inputs` (the text to explain), optionally providing `targets` that specify what to explain (class indices for classification and selected output tokens for generation).
3. Visualize the resulting attributions.

We recommend that users generate the output text they want to explain and pass it to INTERPRETO as `targets`.

```python
import torch
from transformers import AutoTokenizer,
      AutoModelForSequenceClassification
from interpreto import Lime, plot_attributions

# Load the model and tokenizer
repo_id = "nateraw/bert-base-uncased-emotion"
model = AutoModelForSequenceClassification\
    .from_pretrained(repo_id)
tokenizer = AutoTokenizer.from_pretrained(repo_id)
classes_names =\
    ['sadness', 'joy', 'love', 'anger', 'fear', 'surprise']

# Instantiate the explainer
explainer = Lime(model, tokenizer)

# Compute the explanation
attributions = explainer(
    "We are thrilled to present you Interpreto!",
    targets=torch.arange(len(classes_names)).view(1, -1)
)

# Visualize the explanation
plot_attributions(attributions[0],
      classes_names=classes_names)
```

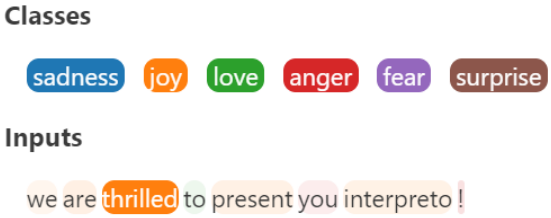


Figure 2: Classification attribution explanation with INTERPRETO. (Top) Minimal code to compute attributions with `Lime` (Ribeiro et al., 2016) for a BERT model (Devlin et al., 2019) fine-tuned on emotion recognition (Saravia et al., 2018). (Bottom) Notebook visualization of the resulting attributions: words are highlighted by importance, and colors correspond to classes. The model predicts the "joy" emotion, mainly driven by "thrilled".

Additional features. In addition to the ten methods above, several attribution explainers' parameters provide useful variants:

- `granularity`: Explanations can be produced at the token, word, or sentence level. Because subword tokens and special tokens can be difficult to interpret, the default is **word**-level.
- `input_x_gradient`: For gradient-based methods, returning the elementwise product of input and gradient is formally justified (Shrikumar et al., 2017); enabled by default.
- `inference_mode`: Choose the output space (logits, softmax, or log softmax). Since softmax couples classes, the default is logits.

Available methods and metrics.

- **Perturbation-based (4)**: KernelSHAP (Lundberg and Lee, 2017), LIME (Ribeiro et al., 2016), Occlusion (Zeiler and Fergus, 2014), and Sobol (Fel et al., 2021).
- **Gradient-based (6)**: GradientSHAP (Lundberg and Lee, 2017), Integrated Gradients (Sundararajan et al., 2017), Saliency (Simonyan et al., 2014), SmoothGrad (Smilkov et al., 2017), SquareGrad (Hooker et al., 2019a), and VarGrad (Hooker et al., 2019b).
- **Faithfulness metrics (2)**: Insertion, Deletion (Petsiuk et al., 2018).

```python
from transformers import AutoTokenizer,
      AutoModelForCausalLM
from interpreto import Occlusion, plot_attributions

# Load the model and tokenizer
repo_id = "Qwen/Qwen3-0.6B"
model = AutoModelForCausalLM.from_pretrained(repo_id)
tokenizer = AutoTokenizer.from_pretrained(repo_id)

# Instantiate the explainer
explainer = Occlusion(model, tokenizer)

# Compute the explanation
attributions = explainer.explain(
    "An interpretability open-source library is great,",
    " but, for HF language models, it is even better.",
)

# Visualize the explanation
plot_attributions(attributions[0])
```

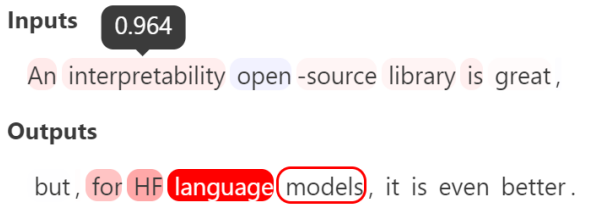


Figure 3: Generation attribution explanation with INTERPRETO. (Top) Minimal code to compute attributions with `Occlusion` (Zeiler and Fergus, 2014) for a Qwen3-0.6B causal language model (Yang et al., 2025) on a hand-crafted input-output pair. (Bottom) Notebook visualization: in generation, each output token is a separate prediction, so attributions usually form an input-output matrix; instead, INTERPRETO lets users select a token of interest. Here, the token "models" is selected (red border), and the most influential input token is "language".

Custom method. INTERPRETO is designed to make new methods lightweight to implement: users write the method-specific computation, while the library handles shared engineering concerns. When existing building blocks are close to the desired behavior, we recommend reusing them and adapting an existing implementation. Internally, the pipeline follows three stages (similar in spirit to the abstraction of Ferré et al. (2025)):

- **Perturbations:** starting from one sample, construct perturbed variants (used by most methods, including some gradient-based ones). For a custom perturbation-based method, users can inherit from the input-IDs perturbator and implement the `mask_function` method; for gradient-based methods, the corresponding component is the embedding perturbator.
- **Inference or gradients:** run forward and/or

backward passes on perturbed samples. This stage typically only needs subclassing to modify the forward or backward computation.

- **Aggregation:** aggregate intermediate scores into element-wise importance scores. For a custom aggregator, users inherit from the base aggregator and implement `aggregate`.

6 `interpreto.concepts`

Definition and taxonomy. Concepts are interpretable computational features used by the model (examples in Figs. 6 and 8b). If users can i) predict a sample's concept activations from the input and ii) predict the output from these activations, they can simulate (and understand) the model's behavior. There are two axes Bhalla et al. (2024):

- **Post hoc vs. by design:** concepts can be introduced during model construction or extracted from a trained model (INTERPRETO positioning).
- **Supervised vs. unsupervised:** Concepts can be specified in advance (*e.g.*, probes (Alain and Bengio, 2016; Belinkov, 2022) or CAVs (Kim et al., 2018)) (planned for later) or discovered from activations. INTERPRETO currently emphasizes unsupervised discovery (dictionary learning).

Concept-based explanations can be **global** (concepts that are important for a class) or **local** (concepts that are active for a specific input or important for the prediction). INTERPRETO supports both; global analyses often provide the basis for local ones. This line of work is connected to mechanistic interpretability (MI); sparse autoencoders (SAEs) are a common approach for concept discovery.

Pipeline. Post-hoc, unsupervised concept methods typically follow four steps (Fel et al., 2023a; Poché et al., 2025). These steps are reflected in Figs. 5 and 8a and summarized in Fig. 4:

1. **Split the model** into a **feature extractor** and a **predictor**, then run a dataset through the extractor to collect activations.
2. **Construct the concept space:** train a concept model on these activations. Each learned concept corresponds to one dimension of the concept space. The concept model maps between the activation space and the concept space.
3. **Interpret concepts:** assign human-meaningful labels (*e.g.*, by linking inputs to concept activations) to each concept dimension.
4. **Estimate concept importance:** quantify each concept's contribution to the predictions.

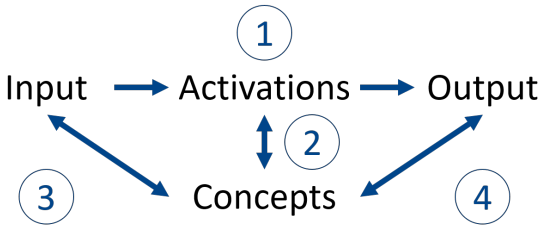


Figure 4: Post-hoc, unsupervised concept-based pipeline. (1) Split the model and extract a dataset of activations. (2) Learn concepts as patterns in these activations. (3) Interpret concepts by assigning human-meaningful labels. (4) Estimate each concept's importance for the model predictions (Sec. 6).

Classification vs. generation. The API is shared across tasks, but recommended settings differ:

- **Activation granularity:** use the [CLS] token for classification and tokens for generation. Granularity is set during activation extraction.
- **Interpretation:** match the interpretation granularity to the activation granularity; the tutorials provide task-specific guidance.

Methods to define the concept space.

- **Neurons as concepts:** treat individual neurons as concepts (Geva et al., 2022).
- **Dictionary learning:** KMeans (Ghorbani et al., 2019), PCA (Zhang et al., 2021), SVD (Graziani et al., 2023; Jourdan et al., 2023a), ICA (Poché et al., 2025), NMF (Zhang et al., 2021; Fel et al., 2023b; Jourdan et al., 2023b), Semi-NMF, Convex NMF (Fel et al., 2025).
- **Sparse autoencoders (SAEs):** Vanilla SAE (Bricken et al., 2023; Huben et al., 2024), JumpReLU SAE (Rajamanoharan et al., 2024), Top-k SAE, Batch Top-k (Gao et al., 2025), Matching Pursuit SAE (Costa et al., 2025).

Methods for concept interpretation. Top-k vocabulary tokens (Geva et al., 2022), top-k activating examples/words/n-grams (MaxAct) (Bricken et al., 2023), and LLM-based labeling (Bills et al., 2023).

Methods to estimate concept importance. Concept-to-output gradients (Fel et al., 2023a) and concept $\times$ gradients (Poché et al., 2025). Compatibility with the attributions module is planned.

Metrics. Evaluate **concept-space** faithfulness via MSE (Bricken et al., 2023) and FID (Fel et al., 2023a); measure sparsity (Bricken et al., 2023) and Stability (Fel et al., 2023a; Paulo and Belrose, 2025); finally, evaluate explanation **general usefulness** via ConSim (Poché et al., 2025).

```python
import datasets
from transformers import AutoModelForCausalLM

from interpreto import ModelWithSplitPoints
from interpreto.concepts import SemiNMFConcepts
from interpreto.concepts.interpretations import LLMLabels
from interpreto.model_wrapping import llm_interface

TOKEN = ModelWithSplitPoints.activation_granularities.TOKEN

# 1.1 Split your model in two parts
mwsp = ModelWithSplitPoints(
    "Qwen/Qwen3-0.6B",
    automodel=AutoModelForCausalLM,
    split_points=[5],
)

# 1.2 Compute a dataset of activations
dataset = datasets.load_dataset(
    "fancyzhx/ag_news")["train"]["text"][:100]
activations = mwsp.get_activations(dataset, TOKEN)

# 2. Train the concept model
explainer = SemiNMFConcepts(mwsp, nb_concepts=20)
explainer.fit(activations)

# 3. Interpret the concepts
llm = llm_interface.OpenAILLM(os.getenv("OPENAI_API_KEY"))
interpreter = LLMLabels(
    concept_explainer=explainer,
    activation_granularity=TOKEN,
    llm_interface=llm,
    k_context=10,
)
interpretations = interpreter.interpret(
    "all", dataset, activations)
```

Figure 5: Generation concept-based code example (steps 1-3). We analyze Qwen3-0.6B (Yang et al., 2025) by learning concepts with Semi-NMF (Fel et al., 2025) from activations computed on 100 AG-News samples (Zhang et al., 2015). We then label concepts with GPT-4.1-nano (OpenAI, 2025) using our default prompting scheme. We omit concept importance, which is less informative for global analyses of generation models. It runs in under 3 minutes on an RTX 3080 (10GB).

```
Proper Noun Indicators
Entity Names
Market Focus
Abbreviation Markers
Named Entity Indicators
Keyword Indicators
Acronym Prefixes
Entity References
Proper Nouns or Named Entities
Quantification or comparison markers
```

Figure 6: First ten concept labels produced by the LLM-based interpreter for the code in Fig. 5. Labels depend on the system prompt; by default, we request short but discriminative labels. Note that the example uses an API to externalize computations, but the labeling can also be done locally. Additionally, concept-labels are hard to tune; they can be vague and redundant or too precise and lengthy; this is discussed in the limitations section 7.

Custom method. Each step is modular, allowing users to mix and match: split points, concept models, and interpretation methods. To add a new method, users implement the corresponding abstraction and focus on the computational core; the library handles integration and execution. For new components, we recommend using existing implementations as templates.

API. The API follows the pipeline steps. Fig. 1b summarizes the module, and Figs. 5 and 8a provide minimal examples; additional examples are available in the classification and generation tutorials.

1. **Split the model** wrap the HuggingFace model with `ModelWithSplitPoints` (built on `nnsight` (Fiotto-Kaufman et al., 2024)). Collect activations with the wrapper.
2. **Construct the concept space:** Collect activations through the split model wrapper. Instantiate an explainer (linked to concept models) with the wrapped model and fit on the activation dataset. Many explainers leverage `overcomplete` (Fel, 2025).
3. **Interpret concepts:** run an interpretation method on the fitted explainer.
4. **Estimate concept importance:** compute concept importance scores (*e.g.*, via the explainer's concept-to-output gradient method).

7 Limitations and outlook

INTERPRETO focuses on HuggingFace text language models and does not cover all interpretability settings. Some limitations are inherent to the explainability methods, and we leave improving the state of the art to other work, including the plausibility-faithfulness trade-off and human biases in interpretation. SAE-based methods can be computationally expensive, and LLM-based interpretations are sensitive to prompt design. Additionally, the library is under active development, so we plan to expand its coverage. Nonetheless, some elements will remain outside the library's scope. Details are provided below.

Limitations of the literature

- There is no single method to govern them all. Users have to apply several methods and compare them using metrics.
- Explanations are subject to human biases (such as confirmation bias and over-interpretations), therefore, they should be treated carefully.
- The meaning of attribution scores depends on the methods. Similar scores from different methods can mean different things.
- LLM-based concepts interpretations are highly sensitive to the given prompt. Depending on the specifications, labels may be too general, preventing us from differentiating among several concepts. Or too specific to be actionable.
- A not interpretable concept can come from a bad

model, a bad concept-space, or a bad interpretation, and we lack ways to determine which one.

Outlook (ongoing development).

- Add supervised concept options (CAVs (Kim et al., 2018) and probes (Alain and Bengio, 2016; Belinkov, 2022)).
- Extend interpretation-specific metrics (*e.g.*, Detection/Fuzzing/Clarity/Purity (Paulo et al., 2025; Puri et al., 2025)).
- Add additional attribution methods (*e.g.*, RISE (Petsiuk et al., 2018), RFEM (Ayyar et al., 2025)) and metrics (*e.g.*, AOPC comprehensiveness and sufficiency).
- Link the two modules to allow input-to-concepts attributions.

Outlook (later). We plan to extend library coverage to ViT (Vision Transformer) to later aim for multi-modal transformer support.

Out of scope. We do not plan to include circuit-level MI methods, data attribution, or feature visualization; the library will remain centered on HuggingFace model workflows.

Ethical statement

Impact. INTERPRETO lowers the barrier to applying attribution and concept-based interpretability methods to HuggingFace language models. Easier access can support model auditing, debugging, and documentation, including the identification of biases and recurring failure modes. At the same time, interpretability outputs can be misread as faithful causal explanations; results depend on method choice, hyperparameters, and presentation. Users should therefore treat explanations as diagnostic evidence rather than ground truth, and corroborate findings with additional tests (e.g., counterfactual checks, ablations, and evaluation on targeted slices).

LLM use. LLM-assisted code suggestions were used during development. Any generated code was manually reviewed by the developer and again during pull-request review before merging. During paper writing, LLMs were only used for editing existing text (e.g., sentence rewriting for clarity) and for flagging missing elements expected in a system demonstration paper; no manuscript text was generated from scratch.

Acknowledgments

Our work has benefited from the AI Cluster ANITI and the research programs DEEL[6] and FOR[7]. ANITI is funded by the France 2030 program under the Grant agreement n°ANR-23-IACL-0002. DEEL and FOR are integrative programs of the AI Cluster ANITI, designed and operated jointly with IRT Saint Exupéry, with the financial support from its industrial and academic partners and the France 2030 program under the Grant agreement n°ANR-10-AIRT-01.

We also acknowledge the support of Pr. Philippe Muller and Dr. Grégory Flandin.

References

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, and 1 others. 2023. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*.

Giuseppe Attanasio, Eliana Pastor, Chiara Di Bonaventura, and Debora Nozza. 2023. ferret: a framework for benchmarking explainers on transformers. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Meghna P Ayyar, Jenny Benois-Pineau, and Akka Zemmari. 2025. There is more to attention: Statistical filtering enhances explanations in vision transformers. *arXiv preprint arXiv:2510.06070*.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*.

Nora Belrose and Lucia Quirke. 2025. Sparsify transformers with saes and transcoders. https://github.com/EleutherAI/sparsify.

Usha Bhalla, Suraj Srinivas, Asma Ghandeharioun, and Himabindu Lakkaraju. 2024. Towards unifying interpretability and control: Evaluation via intervention. *arXiv preprint arXiv:2411.04430*.

Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. 2023. Language models can explain neurons in language models. *OpenAI*.

[6] https://www.deel.ai/

[7] https://www.irt-saintexupery.com/for-program/

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. Gpt-neo: Large scale autoregressive language modeling with mesh-tensorflow. https://doi.org/10.5281/zenodo.5297715.

Joseph Bloom and David Chanin. 2024. Saelens. https://github.com/jbloomAus/SAELens.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Valérie Costa, Thomas Fel, Ekdeep Singh Lubana, Bahareh Tolooshams, and Demba Ba. 2025. From flat to hierarchical: Extracting sparse representations with matching pursuit. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Clément Dumas. 2025. nnterp: A standardized interface for mechanistic interpretability of transformers. In *Mechanistic Interpretability Workshop at NeurIPS 2025*.

Thomas Fel. 2025. Overcomplete: A vision-based sae toolbox. https://github.com/KempnerInstitute/overcomplete.

Thomas Fel, Victor Boutin, Mazda Moayeri, Rémi Cadène, Louis Bethune, Mathieu Chalvidal, Thomas Serre, and 1 others. 2023a. A holistic approach to unifying automatic concept extraction and concept importance estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Thomas Fel, Remi Cadene, Mathieu Chalvidal, Matthieu Cord, David Vigouroux, and Thomas Serre. 2021. Look at the variance! efficient black-box explanations with sobol-based sensitivity analysis. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Thomas Fel, Lucas Hervier, David Vigouroux, Antonin Poche, Justin Plakoo, Remi Cadene, Mathieu Chalvidal, Julien Colin, Thibaut Boissin, Louis Bethune, Agustin Picard, Claire Nicodeme, Laurent Gardes, Gregory Flandin, and Thomas Serre. 2022. Xplique: A deep learning explainability toolbox. *Workshop on Explainable Artificial Intelligence for Computer Vision (CVPR)*.

Thomas Fel, Ekdeep Singh Lubana, Jacob S. Prince, Matthew Kowal, Victor Boutin, Isabel Papadimitriou, Binxu Wang, Martin Wattenberg, Demba Ba, and Talia Konkle. 2025. Archetypal sae: Adaptive and stable dictionary learning for concept extraction in large vision models. In *Proceedings of the International Conference on Machine Learning (ICML)*.

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

Quentin Ferré, Ismail Bachchar, Hakima Arroubat, Aziz Jedidi, Youssef Achenchabe, and Antoine Bonnefoy. 2025. Muppet: A modular and constructive decomposition for perturbation-based explanation methods. https://research.euranova.eu/2025/08/04/muppet-a-modular-and-constructive-decomposition-for-perturbation-based-explanation-methods/.

Jaden Fiotto-Kaufman, Alexander R Loftus, Eric Todd, Jannik Brinkmann, Caden Juang, Koyena Pal, Can Rager, Aaron Mueller, Samuel Marks, Arnab Sen Sharma, Francesca Lucchetti, Michael Ripa, Adam Belfki, Nikhil Prakash, Sumeet Multani, Carla Brodley, Arjun Guha, Jonathan Bell, Byron Wallace, and David Bau. 2024. Nnsight and ndif: Democratizing access to foundation model internals. *arXiv preprint arXiv:2407.14561*.

Leo Gao, Tom Dupré la Tour, Henk Tillman, Gabriel Goh, Rajan Troll, Alec Radford, Ilya Sutskever, Jan Leike, and Jeffrey Wu. 2025. Scaling and evaluating sparse autoencoders. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. 2019. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems (NeurIPS)*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mara Graziani, An-phi Nguyen, Laura O'Mahony, Henning Müller, and Vincent Andrearczyk. 2023. Concept discovery and dataset exploration with singular value decomposition. In *ICLR 2023 Workshop on Pitfalls of limited data and computation for Trustworthy ML*.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. 2019a. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. 2019b. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Robert Huben, Hoagy Cunningham, Logan Riggs Smith, Aidan Ewart, and Lee Sharkey. 2024. Sparse autoencoders find highly interpretable features in language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dongsheng Jiang, Yuchen Liu, Songlin Liu, Jin'e Zhao, Hao Zhang, Zhen Gao, Xiaopeng Zhang, Jin Li, and Hongkai Xiong. 2023. From clip to dino: Visual encoders shout in multi-modal large language models. *arXiv preprint arXiv:2310.08825*.

Sonia Joseph, Praneet Suresh, Lorenz Hufe, Edward Stevinson, Robert Graham, Yash Vadi, Danilo Bzdok, Sebastian Lapuschkin, Lee Sharkey, and Blake Aaron Richards. 2025. Prisma: An open source toolkit for mechanistic interpretability in vision and video. *arXiv preprint arXiv:2504.19475*.

Fanny Jourdan, Louis Béthune, Agustin Picard, Laurent Risser, and Nicholas Asher. 2023a. Taco: Targeted concept erasure prevents non-linear classifiers from detecting protected attributes. *arXiv preprint arXiv:2312.06499*.

Fanny Jourdan, Agustin Picard, Thomas Fel, Laurent Risser, Jean Michel Loubes, and Nicholas Asher. 2023b. Cockatiel: Continuous concept ranked attribution with interpretable elements for explaining neural net classifiers on nlp tasks. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Adam Karvonen, Can Rager, Johnny Lin, Curt Tigges, Joseph Bloom, David Chanin, Yeu-Tong Lau, Eoin Farrell, Callum McDougall, Kola Ayonrinde, and 1 others. 2025. Saebench: A comprehensive benchmark for sparse autoencoders in language model interpretability. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and 1 others. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *Proceedings of the International Conference on Machine Learning (ICML)*.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and 1 others. 2020. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Johnny Lin. 2023. Neuronpedia: Interactive reference and tooling for analyzing neural networks. https://www.neuronpedia.org.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, and 1 others. 2024. Starcoder 2 and the stack v2: The next generation. *arXiv preprint arXiv:2402.19173*.

Scott Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NIPS)*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Callum McDougall. 2024. Sae visualizer. https://github.com/callummcdougall/sae_vis.

Vincent Mussot, François Hoofd, Fanny Jourdan, and Antonin Poché. 2026. Bringing nlp explainability to critical sectors: A case study on notams in aviation. In *Proceedings of the Embedded Real Time Software and Systems Conference (ERTS)*.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. https://github.com/TransformerLensOrg/TransformerLens.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. Codegen: An open large language model for code with multi-turn program synthesis.

In *Proceedings of the International Conference on Learning Representations (ICLR)*.

OpenAI. 2025. Introducing GPT-4.1 in the API. https://openai.com/index/gpt-4-1/.

Gonçalo Paulo and Nora Belrose. 2025. Sparse autoencoders trained on the same data learn different features. *arXiv preprint arXiv:2501.16615*.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. 2025. Automatically interpreting millions of features in large language models. *Proceedings of the International Conference on Machine Learning (ICML)*.

Vitali Petsiuk, Abir Das, and Kate Saenko. 2018. Rise: Randomized input sampling for explanation of black-box models. In *Proceedings of the British Machine Vision Conference (BMVC)*.

Antonin Poché, Alon Jacovi, Agustin Martin Picard, Victor Boutin, and Fanny Jourdan. 2025. Consim: Measuring concept-based explanations' effectiveness with automated simulatability. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Bruno Puri, Aakriti Jain, Elena Golimblevskaia, Patrick Kahardipraja, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. 2025. Fade: Why bad descriptions happen to good features. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *arXiv preprint arXiv:1901.05207*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research (JMLR)*.

Senthooran Rajamanoharan, Tom Lieberum, Nicolas Sonnerat, Arthur Conmy, Vikrant Varma, János Kramár, and Neel Nanda. 2024. Jumping ahead: Improving reconstruction fidelity with jumprelu sparse autoencoders. *arXiv preprint arXiv:2407.14435*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *Advances in Neural Information Processing Systems (NIPS)*.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. Carer: Contextualized affect representations for emotion recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Gabriele Sarti, Nils Feldhus, Ludwig Sickert, Oskar van der Wal, Malvina Nissim, and Arianna Bisazza. 2023. Inseq: An interpretability toolkit for sequence generation models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *Proceedings of the International Conference on Machine Learning (ICML)*.

K Simonyan, A Vedaldi, and A Zisserman. 2014. Deep inside convolutional networks: visualising image classification models and saliency maps. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. In *Workshop on Visualization for Deep Learning*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2025. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. *ArXiv e-print*.

Zhengxuan Wu, Atticus Geiger, Aryaman Arora, Jing Huang, Zheng Wang, Noah Goodman, Christopher Manning, and Christopher Potts. 2024. pyvene: A library for understanding and improving PyTorch models via interventions. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Matthew D Zeiler and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *Proceedings of the IEEE European Conference on Computer Vision (ECCV)*.

Ruihan Zhang, Prashan Madumal, Tim Miller, Krista A Ehinger, and Benjamin IP Rubinstein. 2021. Invertible concept-based explanations for cnn models with non-negative concept activation vectors. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS)*.

A Related libraries details

Attribution libraries. Explainability libraries can be grouped into two broad families: attribution and mechanistic/concept ones. In the first family, LIME (Ribeiro et al., 2016) and SHAP (Lundberg and Lee, 2017) are model- and modality-agnostic toolkits. For NLP, Inseq (Sarti et al., 2023) and Ferret (Attanasio et al., 2023) provide language-focused APIs and visualizations. Captum (Kokhlikyan et al., 2020) offers a broad set of attribution methods in PyTorch but does not include unsupervised concept discovery. Xplique (Fel et al., 2022) spans both attribution and concept tools, with an emphasis on vision rather than NLP.

Mechanistic interpretability libraries. Concept-based explanations are part of mechanistic interpretability (MI). MI libraries target MI researchers and specific stages of the concept pipeline (see Sec. 6). For activation access and intervention, see TransformerLens (Nanda and Bloom, 2022), NNsight (Fiotto-Kaufman et al., 2024), Neuronpedia (Lin, 2023), and NNterp (Dumas, 2025). For training concept models (e.g., SAEs), Sparsify (Belrose and Quirke, 2025), SAELens (Bloom and Chanin, 2024), Overcomplete (Fel, 2025), and ViT-Prisma (Joseph et al., 2025) provide tooling. For interpretation and visualization, Delphi (Paulo et al., 2025) and SAE-Vis (McDougall, 2024) are complementary. Finally, SAEBench (Karvonen et al., 2025), AxBench (Wu et al., 2025), and Pyvene (Wu et al., 2024) aim to evaluate concept-based methods, including via steering.

B Tested models architectures

We test more than 15 model architectures: Albert (Lan et al., 2020); BART (Lewis et al., 2020); BERT (Devlin et al., 2019); DistilBERT (Sanh et al., 2019); Electra (Clark et al., 2020); Roberta (Liu et al., 2019); T5 (Raffel et al., 2020); GPT2 (Radford et al., 2019); GPT-Neo (Black et al., 2021); GPT-J (Black et al., 2021); CodeGen (Nijkamp et al., 2023); Falcon (Almazrouei et al., 2023); Llama3 (Grattafiori et al., 2024); Mistral (Jiang et al., 2023); Starcoder (Lozhkov et al., 2024); Qwen3 (Yang et al., 2025).

C Demonstration website example

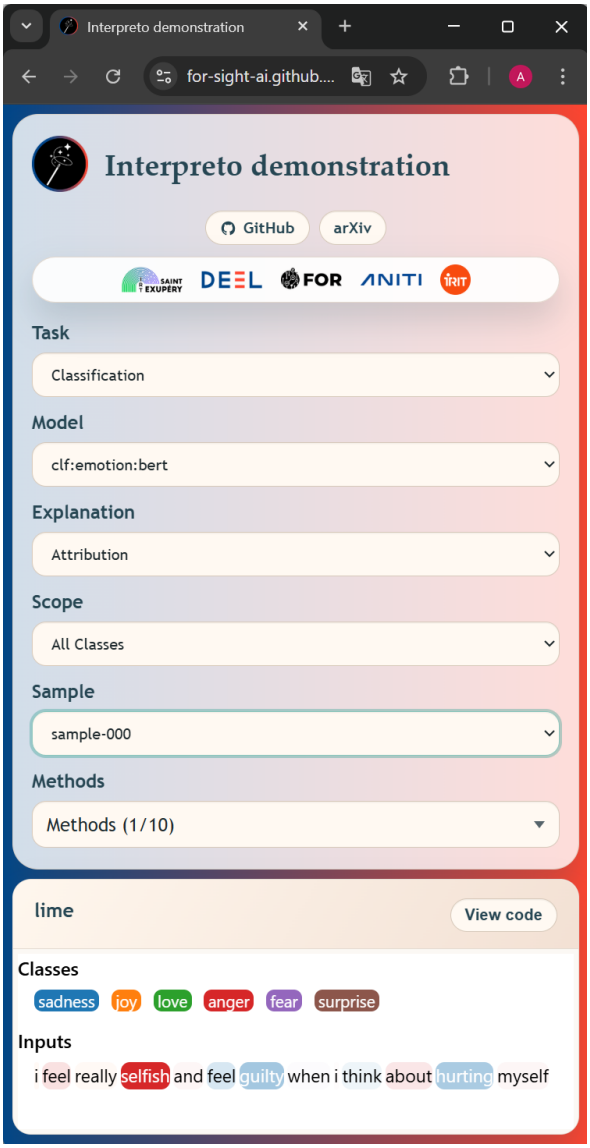


Figure 7: Screenshot from the demonstration website. The user can select a task, model, and explanation format to explore the explanation gallery. Website link.

D Concept-based classification example

```python
import os
from datasets import load_dataset
from transformers import AutoModelForSequenceClassification
from interpreto import ModelWithSplitPoints
from interpreto.concepts import LLMLabels, SemiNMFConcepts
from interpreto.model_wrapping.llm_interface import OpenAILLM

# set the granularity to [CLS] to keep only this token's activation
granularity = ModelWithSplitPoints.activation_granularities.CLS_TOKEN
# set the LLM interface used to generate labels based on the constructed prompts
llm_interface = OpenAILLM(api_key=os.getenv("OPENAI_API_KEY"), model="gpt-4.1-nano")

# ------------------------------------------------------------------------------------------------------------------------
# 1. Split the model into two and get a class-wise dataset of activations
# 1.1 Split the model
model_with_split_points = ModelWithSplitPoints(
    "textattack/distilbert-base-uncased-ag-news", split_points=[5],  # split at the sixth layer
    automodel=AutoModelForSequenceClassification, device_map="cuda", batch_size=1024
)
# 1.2 Construct the dataset of activations (extract the ones related to the class)
# load the AG-News dataset (here we use only 1000 examples to go faster, but the more, the better)
inputs = load_dataset("fancyzhx/ag_news")["train"]["text"][:1000]
# Compute the [CLS] token activations
activations = model_with_split_points.get_activations(inputs, granularity, include_predicted_classes=True)

for target, class_name in enumerate(classes_names):  # iterate over classes
    # 1.3 Take the subset for the target class (predicted class)
    indices = (activations["predictions"] == target).nonzero(as_tuple=True)[0]
    class_wise_inputs = [inputs[i] for i in indices]
    class_wise_activations = {k: v[indices] for k, v in activations.items()}

    # ------------------------------------------------------------------------------------------------------------------------
    # 2. train concept model
    concept_explainer = SemiNMFConcepts(model_with_split_points, nb_concepts=20, device="cuda")
    concept_explainer.fit(class_wise_activations)

    # ------------------------------------------------------------------------------------------------------------------------
    # 4. compute concepts importance (before interpretations to limit the number of concepts interpreted)
    gradients = concept_explainer.concept_output_gradient(class_wise_inputs, [target],
                                                          activation_granularity=granularity, batch_size=64)
    # stack gradients on samples and average them over samples
    concept_importances = torch.stack(gradients, axis=0).squeeze().abs().mean(dim=0)  # (num_concepts,)
    # for each class, sort the importance scores
    important_concept_indices = torch.argsort(concept_importances, descending=True).tolist()

    # ------------------------------------------------------------------------------------------------------------------------
    # 3. interpret the important concepts concepts
    llm_labels_method = LLMLabels(concept_explainer=concept_explainer, activation_granularity=granularity,
                                  llm_interface=llm_interface, k_examples=20,)
    concept_interpretations = llm_labels_method.interpret(inputs=class_wise_inputs,
                                                          concepts_indices=important_concept_indices)
```

(a) Classification concept-based explanation code example. We explain a DistilBERT (Sanh et al., 2019) classifier fine-tuned on AG News (Zhang et al., 2015). We decompose the concepts with the Semi-NMF (Fel et al., 2025) and interpret them with the LLM labels (Bills et al., 2023) from GPT-4.1-nano (OpenAI, 2025). The concepts are computed for each predicted class.

```
Class: World
        importance: 0.088,      Diverse topics unified by factual reporting and event-focused language
        importance: 0.086,      Consistent focus on named entities, events, and quotations.
        importance: 0.084,      Consistently presents factual info with emphasis on notable events or figures.
        importance: 0.082,      Recurring topics and formal reporting language.
        importance: 0.078,      Concise reporting style with specific names, events, and dates.

Class: Sports
        importance: 0.112,      Patterns involve sports, achievements, and event summaries with emphasis on names, scores...
        importance: 0.105,      Structured sports and news summaries emphasize game events, scores, and highlights, often...
        importance: 0.082,      Focus on key entities, events, and record-breaking achievements.
        importance: 0.076,      Consistent references to sports events, players, and results.
        importance: 0.07,       Concise patterns: factual summaries with focus on specific event outcomes.

Class: Business
        importance: 0.092,      Concise patterns involve financial, infrastructural, and cultural references.
        importance: 0.083,      Structured information emphasis, financial and infrastructural keywords.
        importance: 0.081,      Topical focus, political or economic narratives, and authoritative tone.
        importance: 0.074,      Concise patterns include focus on current events, economic issues, and industry developments.
        importance: 0.072,      Focus on comparative categories and rankings across regions.

Class: Sci/Tech
        importance: 0.121,      Language features that explicitly reference entities, events, or dates.
        importance: 0.101,      Consistent focus on environmental impacts, species, and conservation initiatives.
        importance: 0.095,      Scientific observations of natural phenomena and technological descriptions.
        importance: 0.093,      Factual reporting with scientific, environmental, and technological focus, structured as...
        importance: 0.081,      Factual, technical, and headline-like language with specific details
```

(b) List of class-wise concept labels with their global importance for the class.

Figure 8: Code and output for a global classification concept-based explanations. Here, concepts are computed class-wise; it is not mandatory, but it gives better, more specific concepts. The model is a DistilBERT (Sanh et al., 2019) classifier fine-tuned on the AG News (Zhang et al., 2015) dataset. The code and outputs are extracted from the classification concepts tutorial.