LM Transparency Tool: Interactive Tool for Analyzing Transformer Language Models

Igor Tufanov[∞] **Karen Hambardzumyan**[∞□] **Javier Ferrando**[◇*] **Elena Voita**[∞]
[∞]AI at Meta (FAIR) [□]University College London [◇]Universitat Politècnica de Catalunya
{igortufanov,mahnerak,lenavoita}@meta.com
javier.ferrando.monsonis@upc.edu

Abstract

We present the LM Transparency Tool (LM-TT), an open-source interactive toolkit for analyzing the internal workings of Transformer-based language models. Differently from previously existing tools that focus on isolated parts of the decision-making process, our framework is designed to make the entire prediction process transparent, and allows tracing back model behavior from the top-layer representation to very fine-grained parts of the model. Specifically, it (i) shows the important part of the whole input-to-output information flow, (ii) allows attributing any changes done by a model block to individual attention heads and feed-forward neurons, (iii) allows interpreting the functions of those heads or neurons. A crucial part of this pipeline is showing the importance of specific model components at each step. As a result, we are able to look at the roles of model components only in cases where they are important for a prediction. Since knowing which components should be inspected is key for analyzing large models where the number of these components is extremely high, we believe our tool will greatly support the interpretability community both in research settings and in practical applications. The LM-TT codebase is available at https://github.com/facebookresearch/llm-transparency-tool.

1 Introduction

Recent advances in natural language processing led to remarkable capabilities of the Transformer language models, especially with scale (Brown et al., 2020; Kaplan et al., 2020; Zhang et al., 2022a; Wei et al., 2022; Ouyang et al., 2022; OpenAI, 2023; Anil et al., 2023; Touvron et al., 2023a,b). This, along with the wide adoption of such models in high-stakes settings, makes understanding the internal workings of these models vital from the safety, reliability and trustworthiness perspectives.

Existing tools for analyzing sequence models' predictions enable users to compute input tokens attribution scores, read token promotions performed by different model components, or analyze textual patterns responsible for the activation of model's neurons (Geva et al., 2022a; Katz and Belinkov, 2023; Alammar, 2021; Tenney et al., 2020; Sarti et al., 2023; Kokhlikyan et al., 2020; Miglani et al., 2023). However, these focus only on specific parts of the decision-making process and none of them is designed to make the entire prediction process transparent. In contrast, we introduce LM Transparency Tool, a framework that allows tracing back model behavior to very fine-grained model parts.

One of the key advantages of our pipeline is the ability to look only at those model components that were relevant for a selected prediction. Indeed, e.g. syntactic attention heads (Voita et al., 2019), induction heads (Elhage et al., 2021; Olsson et al., 2022), knowledge neurons (Dai et al., 2022), etc. perform their function only in specific cases and are "dormant" otherwise – therefore, looking at them makes sense only for those certain examples. To make this possible, our tool first shows the information flow routes introduced by Ferrando and Voita (2024): this is a subset of intermediate token representations and model components that together form the most important part of the entire input-to-output processing. Then, the tool further allows (i) attributing any changes done by those important model blocks to individual attention heads and feed-forward neurons, as well as (ii) interpreting the functions of those heads and neurons. Importantly, LM-TT is highly efficient: due to relying on Ferrando and Voita (2024), it is 100 times faster than typical patching-based alternatives (Conmy et al., 2023).

Overall, the LM Transparency Tool:

- visualizes the "important" part of the prediction process along with importances of model components at varying levels of granularity;

[*] Work done during an internship at Meta.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 51–60

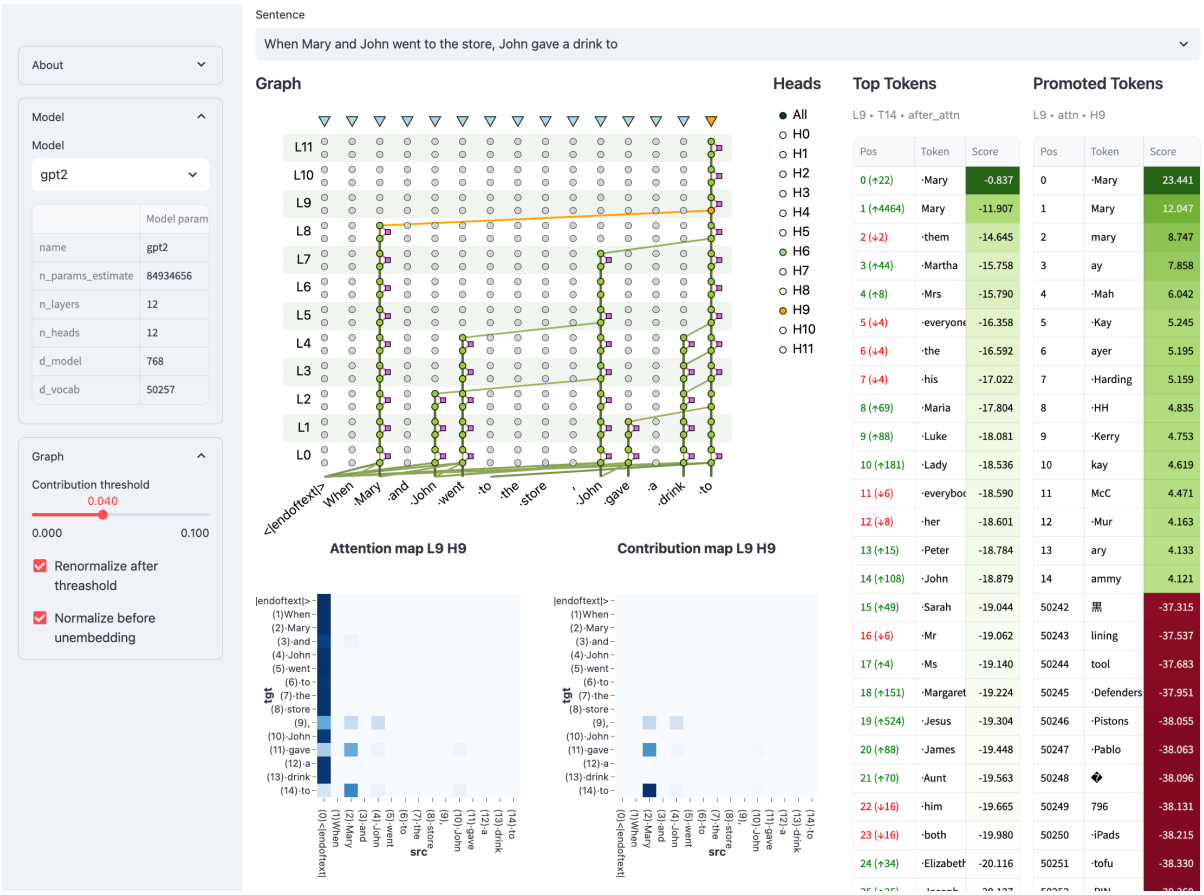

Figure 1: The LM Transparency Tool UI showing information flow graph for the selected prediction, importances of attention heads at the selected layer, attention and contribution maps, logit lens for the selected representation, and top tokens promoted/suppressed by the selected attention head.

- allows interpreting representations and updates coming from model components;
- enables analyzing large models where it is crucial to know what to inspect;
- allows interactive exploration via a UI[1];
- is highly efficient.

2 User Interface and Functionality

Inside Transformer language models, each representation evolves from the current input token embedding[2] to the final representation used to predict the next token. This evolution happens through additive updates coming from attention and feed-forward blocks. The resulting stack of same-token representations is usually referred to as "residual stream" (Elhage et al., 2021), and the overall computation inside the model can be viewed as a sequence of residual streams connected through layer blocks. Formally, we can see it as a graph where nodes correspond to token representations and edges correspond to operations inside the model (attention heads, feed-forward layers, etc.). Our tool visualizes the "important" part of this graph, importances of model components at varying levels of granularity (individual heads and neurons), as well as an interpretation of representations and updates coming from model components.

2.1 Important Information Flow Subgraph

As we mentioned, we can see computations inside the Transformer as a graph with token representations as nodes and operations inside the model as edges. While during model computation all the edges (i.e., model components) are present, computations important for each prediction are likely to form only a small portion of the original graph (Voita et al. (2019); Wang et al. (2023); Hanna et al. (2023), among others). Recent work by Ferrando and Voita (2024) extracts

[1]We also host a demo at https://huggingface.co/spaces/facebook/llm-transparency-tool-demo

[2]Sometimes, along with positional encoding.

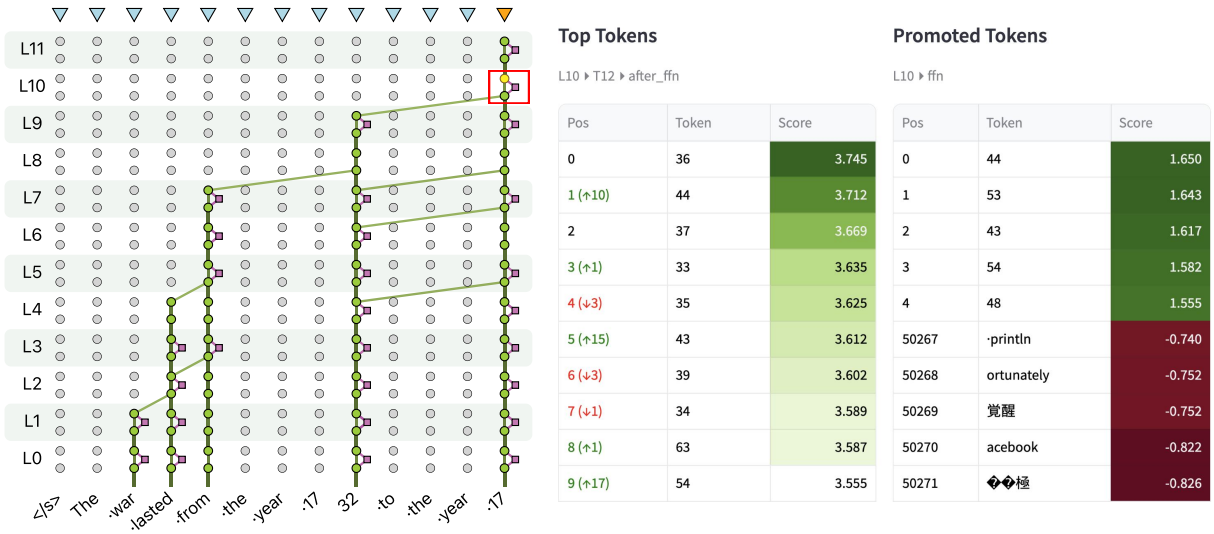

Figure 2: The tool shows how the MLP block on Layer 10 promotes tokens greater than 32, causing the prediction of the end of the war year to be later than the beginning in 1732. Model: OPT-125m (Zhang et al., 2022b).

this important subgraph in a top-down manner by tracing information back through the network and, at each step, leaving only edges that were important (Figure 1). To understand which edges are important, they rely on an attribution method (Ferrando et al., 2022). Ferrando and Voita (2024) explain the benefits of this method including, among other things, why it is more versatile, informative and around 100 times more efficient compared to commonly used patching-based approaches typical for the existing mechanistic interpretability workflows (Wang et al., 2023; Hanna et al., 2023; Conmy et al., 2023; Stolfo et al., 2023; Heimersheim and Janiak, 2023).

In the tool. In the tool, we show only the important attention edges and feed-forward blocks (purple squares in Figure 1). Clicking at the top triangles gives the important information flow routes for each token position. Under the "Graph" menu, one can vary the importance threshold to get more or less dense graphs.

2.2 Fine-Grained Importances

While the information flow graph already relies on the importances of attention or feed-forward blocks for the current residual stream, the tool goes further and shows the importances of (i) individual attention heads, and (ii) individual FFN neurons.

2.2.1 Individual Attention Heads

Importance. After clicking on an attention edge (green lines in Figure 1), the tool shows which specific attention head is mostly responsible for this connection, as well as highlights the importances of other heads for this specific step.

Weights and contributions. Whenever a head is selected, the tool shows

- attention map,
- contribution map.

While the attention map can give an idea of the attention head's function (Voita et al., 2019; Clark et al., 2019; Correia et al., 2019), attention weights might not reflect influences properly (Bastings and Filippova (2020); Kobayashi et al. (2020), among others). Therefore, we also show *contribution map* reflecting the influence of a head-token pair in the overall attention block (Ferrando and Voita, 2024). Note that while attention weights always sum to 1, contributions sum to the overall importance of this attention head at each step. As a result, contribution maps can be more sparse, as shown in Figure 1.

2.2.2 Individual FFN Neurons

When clicking on feed-forward blocks (purple squares), the tool shows the top neurons that contributed at this step. Note that this is different from previous work that either considered top activated neurons (Geva et al. (2022a); Alammar (2021), among others) or did not consider neurons at all (Tenney et al. (2020); Sarti et al. (2023), among others). In contrast, our tool shows top *contributing* neurons and makes it possible to look at the functions of neurons only when they are important, i.e. when they perform their function.

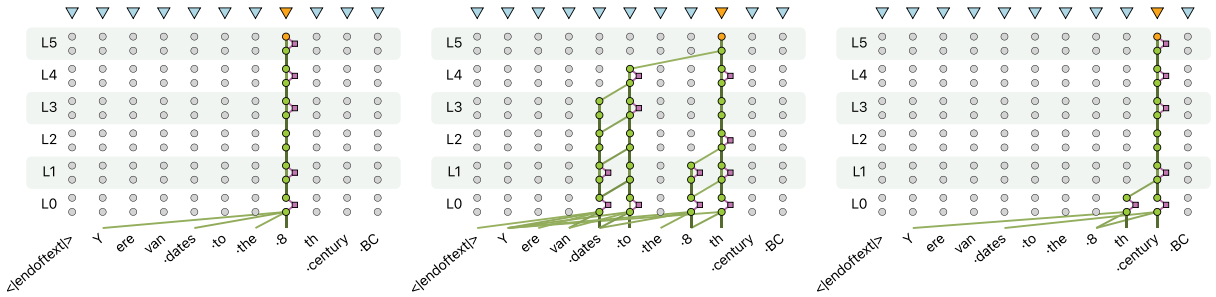

Figure 3: The tool efficiently precomputes the information flow routes across all predictions with a single pass. Clicking on triangle buttons switches between token predictions on different positions. One can see that information flow routes are rather wide for some predictions and narrow for the others. Model: DistilGPT2 (Sanh et al., 2019).

2.3 Vocabulary Projections

One of the popular ways to interpret vector representations is to project them onto the model's vocabulary space. Our tool does this for (i) representations in the residual stream and (ii) the updates coming from specific model components.

2.3.1 Interpreting Representations

While to get a prediction, we project the final-layer representation onto the output vocabulary, for interpretation, we can project representations at any point inside the residual stream – this is called *logit lens* (nostalgebraist, 2020). The resulting sequence of distributions (or top-token predictions) illustrates the decision-making process over the course of the Transformer inference. This is used rather prominently to trace the bottom-up changes in the residual stream (Alammar (2021); Geva et al. (2021, 2022b); Merullo et al. (2023); Belrose et al. (2023); Din et al. (2023), among others).

Tool: click on the circles. In our tool, circles correspond to residual stream representations after applying each model block, either attention or feed-forward; overall, we have two representations per layer. By clicking at each circle, under "Top tokens" the tool shows the projection of this residual state onto output vocabulary.[3]

2.3.2 Interpreting Model Components

We can also project onto vocabulary an update coming from a model component: this shows how this component changes the residual stream and, therefore, gives an interpretation of its behavior. In this way, we can get concepts *promoted* by this component by looking at top positive projections (Geva et al. (2022b); Dar et al. (2023), inter alia) or *suppressed concepts* by looking at bottom negative projections (Voita et al., 2024). Figures 2 and 4 show examples of such cases.

Tool: click on the circles and go further. When you click on a representation from the residual stream, in addition to this representation's logit lens, the tool will also show top promoted and suppressed concepts for the last applied block (either attention or feed-forward). By clicking further, you can also select an individual attention head or feed-forward neuron and get an interpretation at a finer-grained level.

2.4 Additional Controls

For the functionality above, the sidebar to the left has additional controls:

- **Model**:
  - `GPT-2` (Radford et al., 2019),
  - `OPT` (Zhang et al., 2022b),
  - `Llama-2` (Touvron et al., 2023b),
  - ◇ add your own model (Section 3.6);
- **Device**: `GPU` or `CPU`;
- **Data**: adding custom data or choosing an existing example;
- **Graph**: tuning parameters of the information flow graph, e.g. contribution threshold etc. (Ferrando and Voita, 2024).

2.5 Intended Use Cases

The tool can help generating or validating hypotheses about model functioning more quickly. The list of potential use cases contains, but is not limited to the following:

[3] Under "Graph", one also specifies whether to apply the final layer normalization before projecting onto vocabulary or not.

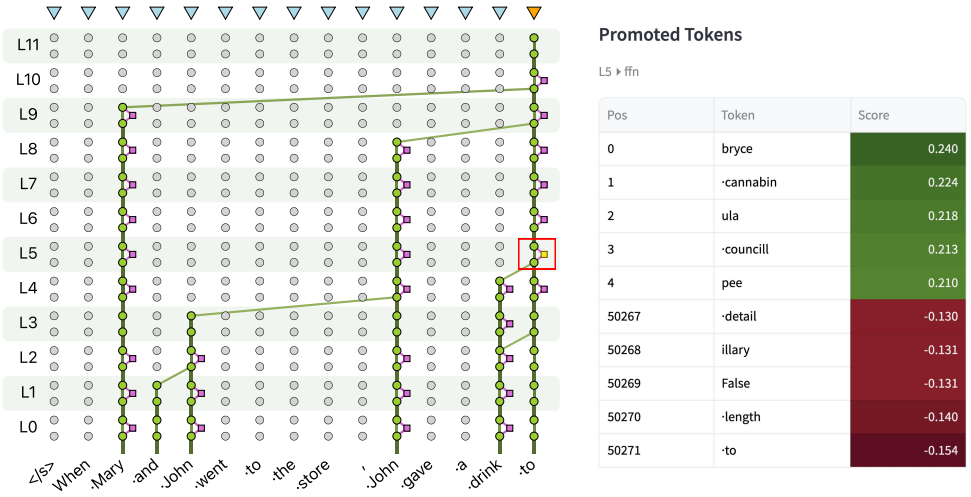

Figure 4: The input token is being suppressed by the neurons in MLP block at Layer 5, similar to the findings reported by Voita et al. (2024). Model: OPT-125m (Zhang et al., 2022b).

- finding model components amplifying biases;
- checking whether the model is reasoning via different routes for desired/undesired behavior (e.g., in safety settings);
- validating whether e.g. mathematical tasks are solved via computation rather than memorization;
- inspecting model behavior for factuality, when hallucinating, etc.

3 System Design and Components

Our application is a web-based toolkit, offering easy and interactive access that is cross-platform compatible. This approach allows users to utilize and share the tool remotely, emphasizing convenience and flexibility.

3.1 Frontend

The frontend is developed using `Streamlit` (Teixeira et al.), with an additional custom component specifically created for visualization of the Transformer model in the form of a graph. This enhancement was necessary as such complex visualizations are not natively supported by `Streamlit`'s built-in features. The custom component is built using `D3.js` (Bostock et al., 2011) and integrated with `React` for managing dynamic content and user interactions.

3.2 Backend

Our backend is a single-dispatch, stateless `Streamlit` program. It includes a caching mechanism to optimize performance for repeated queries. The modeling and tokenization are powered by Hugging Face `transformers` (Wolf et al., 2020) library. For capturing model activations and intermediate computations, we use `TransformerLens` (Nanda and Bloom, 2022)[4] library as it has hooked wrappers defined for a variety of models.

3.3 Configuration and Deployment

Configuration is handled via a `JSON` file, allowing for customization of parameters such as dataset file access, maximum user string length, the list of available models, a default model and a dataset. An example configuration is shown in Figure 5.

```
{
  "max_user_string_length": 100,
  "preloaded_dataset_filename": "samples.txt",
  "debug": false,
  "models": {
    "facebook/opt-6.7b": "facebook/opt-6.7b",
    "my_gpt": "./local/path/to/my_gpt"
  }
}
```

Figure 5: An example configuration.

In this configuration, model names can be either Hugging Face model identifiers or local paths. Other settings, such as threshold adjustments and computation precision, are directly configurable within the application's user interface, enabling quick switching.

Overall, launching the tool is as easy as:

[4] https://github.com/neelnanda-io/TransformerLens

```
streamlit run app.py -- path/to/config.json
```

3.4 Computations

For a selected sentence and model, the tool makes the forward pass and uses the following tensors:

- intermediate representations: residual stream states before and after each block;
- each block's output: the value added to the residual stream by FFN or attention;
- attention block internal states: attention weights, per-head block output, token-specific terms in each head's output;
- FFN block internal states: neuron activations before and after the activation function.

Using this, the tool computes the importances of all the elements (blocks, heads, neurons) and extracts the information flow graph (Ferrando and Voita, 2024). Vocabulary projections and importances within a layer are done on-the-fly when a user clicks on an element.

Supported model sizes. We tested the tool with models up to `30b` of parameters. Since for simplicity and ease of debugging we focus on single-node setup, larger models requiring distributed mode might not work in the current version.

Efficiency. The tool supports automatic mixed precision (`float16` and `bfloat16`). This helps to store model parameters and tensors efficiently, thus saving memory in order to accommodate larger models without sacrificing performance. Models are loaded on demand and cached for efficiency.

3.5 Outside of the UI

Outside of the UI, one can access the underlying functionality of the tool programmatically with Python function calls. For example, getting information flow routes requires the following call:

```
import llm_transparency_tool as lmtt
from lmtt.models.tlens_model import (
    TransformerLensTransparentLlm,
)

model = TransformerLensTransparentLlm(name)

model.run([sentence])

graph = lmtt.routes.graph.build_full_graph(
    model,
    threshold=threshold,
)
```

3.6 Adding Your Own Models

By default, upon installation, the tool supports only the models listed in Section 2.4. Steps needed for adding a new model depend on whether the model is supported by `TransformerLens`.

Supported by TransformerLens. Adding a model supported by `TransformerLens` model is very simple.

- **Hugging Face weights:** Add model name (as stated in Hugging Face `transformers`) to the app's configuration `JSON` file.
- **Custom weights:** In the `JSON` configuration file, along with the name of the model, provide the path to the model file.

Not supported by TransformerLens. In this case, you need to let the tool know how to create proper hooks for the model. Our tool is using `TransformerLens` through an intermediate interface (`TransparentLlm` class) and you have to implement this interface for your model.

4 Related Work

Existing tools for analyzing sequence models' predictions include `LM-Debugger` (Geva et al., 2022a), `VISIT` (Katz and Belinkov, 2023), `Ecco` (Alammar, 2021), `LIT` (Tenney et al., 2020), `Inseq` (Sarti et al., 2023), and `Captum` (Kokhlikyan et al., 2020; Miglani et al., 2023). These tools enable users to compute input tokens attribution scores, read token promotions performed by different model components via logit lens, or analyze textual patterns responsible for the activation of the model's neurons. However, these are not able to extract a relevant part of model computations and indicate component importances. To identify parts of the model relevant for some task, a recent trend in mechanistic interpretability is to rely on causal interventions on the computational graph of the model, aka "activation patching" (Vig et al., 2020; Geiger et al., 2020, 2021; Wang et al., 2023; Hanna et al., 2023; Conmy et al., 2023; Stolfo et al., 2023; Heimersheim and Janiak, 2023). Usually, this process involves the following steps: 1) selecting a dataset and metric, 2) manually creating contrastive examples, 3) searching for important edges in the graph via activation patching. The latter requires running a forward pass per each patched element and uses many patches to explain a single prediction. Although recent approaches aim to automate some

parts of this workflow (Conmy et al., 2023), the entire process requires a large human effort and involves significant computational costs: this imposes constraints on the tool development and limits its applicability. Differently, `LM-TT` relies on a recent method by Ferrando and Voita (2024) which refuses from the patching constraints by relying on attribution to define the importances. Furthermore, `LM-TT` incorporates additional functionalities such as showing fine-grained component importances, logit lens analysis at different levels of granularity, and attention visualization not only via attention weights but also via contributions. This enables users to gain a more comprehensive understanding of the functions executed by each component.

5 Conclusions

We release the LM Transparency Tool, an open-source toolkit for analyzing Transformer-based language models that allows tracing back model behavior to specific parts of the model. Specifically, it (i) shows the important part of the whole input-to-output information flow, (ii) allows attributing any changes done by a model block to individual attention heads and feed-forward neurons, (iii) allows interpreting the functions of those heads or neurons. Notably, due to the nature of the underlying method, our tool reduces the number of components to be analyzed by highlighting model components that were relevant to the prediction. This greatly simplifies the study of large language models, with potentially thousands of attention heads and hundreds of thousands of neurons to look at. Moreover, the UI accelerates the inspection process, unlike other frameworks that lack this feature. This assists researchers and practitioners in efficiently generating hypotheses regarding the behavior of the model.

6 Acknowledgments

We would like to thank Christoforos Nalmpantis, Nicola Cancedda, Yihong Chen, Andrey Gromov and Mostafa Elhoushi for the insightful discussions.

References

J Alammar. 2021. Ecco: An open source library for the explainability of transformer language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 249–257, Online. Association for Computational Linguistics.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. 2023. Palm 2 technical report.

Jasmijn Bastings and Katja Filippova. 2020. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 149–155, Online. Association for Computational Linguistics.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens.

Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss,

Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Arthur Conmy, Augustine N. Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2023. Analyzing transformers in embedding space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16124–16170, Toronto, Canada. Association for Computational Linguistics.

Alexander Yom Din, Taelin Karidi, Leshem Choshen, and Mor Geva. 2023. Jump to conclusions: Short-cutting transformers with linear transformations.

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

Javier Ferrando, Gerard I. Gállego, and Marta R. Costa-jussà. 2022. Measuring the mixing of contextual information in the transformer. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8698–8714, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Javier Ferrando and Elena Voita. 2024. Information flow routes: Automatically interpreting language models at scale.

Atticus Geiger, Hanson Lu, Thomas F Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*.

Atticus Geiger, Kyle Richardson, and Christopher Potts. 2020. Neural natural language inference models partially embed theories of lexical entailment and negation. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 163–173, Online. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022a. LM-debugger: An interactive tool for inspection and intervention in transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–21, Abu Dhabi, UAE. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022b. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model.

Stefan Heimersheim and Jett Janiak. 2023. The singular value decompositions of transformer weight matrices are highly interpretable.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Shahar Katz and Yonatan Belinkov. 2023. VISIT: Visualizing and interpreting the semantic information flow of transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14094–14113, Singapore. Association for Computational Linguistics.

Goro Kobayashi, Tatsuki Kuribayashi, Sho Yokoi, and Kentaro Inui. 2020. Attention is not only a weight: Analyzing transformers with vector norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7057–7075, Online. Association for Computational Linguistics.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. 2023. A mechanism for solving relational tasks in transformer language models.

Vivek Miglani, Aobo Yang, Aram Markosyan, Diego Garcia-Olano, and Narine Kokhlikyan. 2023. Using captum to explain generative language models. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 165–173, Singapore, Singapore. Empirical Methods in Natural Language Processing.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/neelnanda-io/TransformerLens`.

nostalgebraist. 2020. Interpreting gpt: The logit lens.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Alec Radford, Jeff Wu, Rewon Child, D. Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Gabriele Sarti, Nils Feldhus, Ludwig Sickert, and Oskar van der Wal. 2023. Inseq: An interpretability toolkit for sequence generation models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 421–435, Toronto, Canada. Association for Computational Linguistics.

Alessandro Stolfo, Yonatan Belinkov, and Mrinmaya Sachan. 2023. Understanding arithmetic reasoning in language models using causal mediation analysis.

Thiago Teixeira, Amanda Kelly, Adrien Treuille, and Streamlit Team. Streamlit: A faster way to build and share data apps. `https://streamlit.io/`.

Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, and Ann Yuan. 2020. The language interpretability tool: Extensible, interactive visualizations and analysis for NLP models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 107–118, Online. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

Elena Voita, Javier Ferrando, and Christoforos Nalmpantis. 2024. Neurons in large language models: Dead, n-gram, positional. In *Findings of the Association for Computational Linguistics: ACL 2024*, Bangkok, Thailand. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022a. Opt: Open pre-trained transformer language models.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022b. Opt: Open pre-trained transformer language models.