N2G: A SCALABLE APPROACH FOR QUANTIFYING INTERPRETABLE NEURON REPRESENTATION IN LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding the function of individual neurons within language models is essential for mechanistic interpretability research. We propose **Neuron to Graph (N2G)**, a tool which takes a neuron and its dataset examples, and automatically distills the neuron's behaviour on those examples to an interpretable graph. This presents a less labour intensive approach to interpreting neurons than current manual methods, that will better scale these methods to LLMs. We use truncation and saliency methods to only present the important tokens, and augment the dataset examples with more diverse samples to better capture the extent of neuron behaviour. These graphs can be visualised to aid manual interpretation by researchers, but can also output token activations on text to compare to the neuron's ground truth activations for automatic validation. N2G represents a step towards scalable interpretability methods by allowing us to convert neurons in an LLM to interpretable representations of measurable quality.

1 INTRODUCTION

Interpretability of machine learning models is an active research topic (Hendrycks et al., 2021; Amodei et al., 2016) and can have a wide range of applications from bias detection (Vig et al., 2020) to autonomous vehicles (Barez et al., 2022) and Large Language Models (LLMs; Elhage et al. (2022a)). The growing subfield of mechanistic interpretability aims to understand the behaviour of individual neurons within models as well as how they combine into larger circuits of neurons that perform a particular function (Olah et al., 2020; Olah, 2022; Goh et al., 2021), with the ultimate aim of decomposing a model into interpretable components and using this to ensure model safety.

Interpretability tools for understanding neuron in LLMs are lacking. Currently, researchers often look at dataset examples containing tokens on which a neuron strongly activates and investigate common elements and themes across examples to give some insight into neuron behaviour (Elhage et al., 2022a; Geva et al., 2020). However, this can give the illusion of interpretability when real behaviour is more complex (Bolukbasi et al., 2021a), and measuring the degree to which these insights are correct is challenging. Additionally, inspecting individual neurons by hand is time-consuming and unlikely to scale to entire models.

To overcome these challenges, we present **Neuron to Graph (N2G)**, which automatically converts a target neuron within an LLM to an interpretable graph that visualises the contexts in which the neuron activates. The graph can be visualised to facilitate understanding the neuron's behaviour, as well as used to process text and produce predicted token activations. This allows us to measure the correspondence between the target neuron's activations and the graph's activations, which provides a direct measurement of the degree to which a graph captures the neuron's behaviour.

Our method takes maximally activating dataset examples for a target neuron, prunes them to remove irrelevant context, identifies the tokens which are important for neuron activation, and creates additional examples by replacing the important tokens with other likely substitutes using BERT (Devlin et al., 2018). These processed examples are then given as input to the graph builder, which removes unimportant tokens and creates a condensed representation in the form of a trie. This trie can then be used to process text and will predict activations for each token, and can be converted to a graph for visualisation.

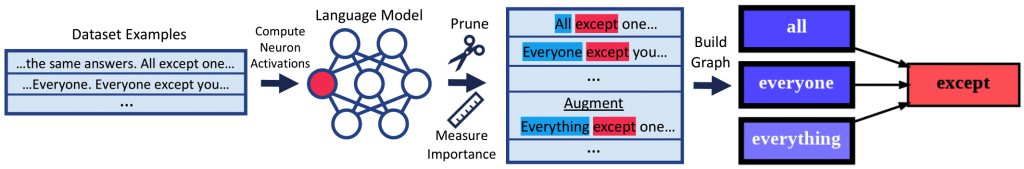

Figure 1: **Overall architecture of N2G.** Activations of the target neuron on the dataset examples are retrieved (neuron and activating tokens in red). Prompts are pruned and the importance of each token for neuron activation is measured (important tokens in blue). Pruned prompts are augmented by replacing important tokens with high-probability substitutes using BERT. The augmented set of prompts are converted to a graph. The output graph is a real example which activates on the token "except" when preceded by any of the other tokens.

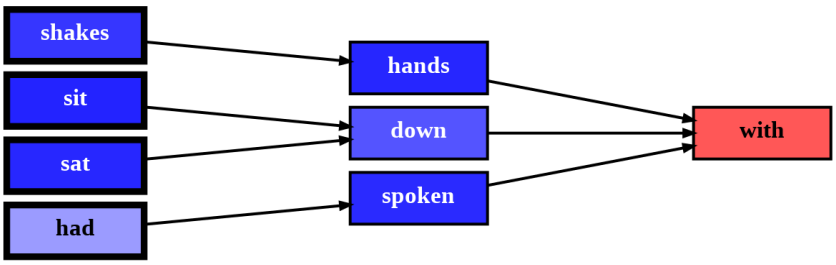

Figure 2: An example of a graph built from Neuron 2 of Layer 1 of the model.

2 METHODOLOGY

N2G constructs an interpretable graph representing the context required for a given neuron to activate on a particular token. The graph can be used to process text and predict whether the target neuron will fire (strongly activate) for each token in the input. N2G takes as input a model, the layer and neuron indices of the target neuron, and a set of prompts which contain one or more tokens for which the target neuron strongly activates. Figure 1 illustrates the overall process as well as an example of a real graph. Figure 2 shows another example and Appendix A.2 contains further examples with interesting behaviours.

Prune: Given a prompt, we process it with the model and retrieve the token activations of the target neuron using the TransformerLens library (Nanda, 2022b), which allows easy access to internal neuron activations. The prune function takes the prompt and activations and finds the token with the highest activation (the key token). It removes all sentences after the key token (we study autoregressive models so these cannot affect neuron activation) and removes all tokens before the key token. It then measures the activation of the neuron on the key token in the truncated prompt to determine the change in activation. If this activation has decreased by more than a user-defined percentage (we choose 50%), then the prior token is added to the truncated prompt. This process is then repeated until the neuron activation on the key token is sufficient to pass the condition.

Saliency: The importance of each token for neuron activation on every other token is then computed to create a matrix of token importance. The importance I_k of the kth token relative to the jth token is calculated as $I_k = 1 - (a_{j,masked}/a_j)$, where a_j is the activation of the neuron on the jth token and $a_{j,masked}$ is the activation of the neuron on the jth token when token k is masked with a special padding token. This method is similar to other perturbation-based saliency methods in Computer Vision (Dabkowski & Gal, 2017) and NLP (Liu et al., 2018).

Augment: The pruned prompt is then used to generate more varied inputs to better explore the neuron's behaviour. Each token that is important for activation on the key token is masked in turn, and BERT (Devlin et al., 2018) predicts the top n substitutions for the masked token. A new prompt is created for each substitute token, provided they cross a probability threshold. This technique is very similar to existing methods of data augmentation used during training (Ma, 2019).

Graph Building: The pruned prompts and the augmented prompts are the input to the graph-building stage, along with normalised token activations and the importance matrix for each prompt. The normalised activation a_N of the ith token is calculated as $a_N = a_i/a_{max}$, where a_{max} is the maximum activation of the neuron on any token in the training dataset.

Each neuron graph is implemented as a trie, with each node representing a token. The first layer of nodes contains tokens on which the neuron strongly activates, and each sub-trie of one of these activating nodes represents the contexts for which the neuron will activate on that token.

For every token in the prompt with a normalised activation above a threshold, we create a top layer node in the trie. Starting at the given activating token, we work backwards through the preceding tokens, adding them to the trie if they have an importance for the activating token above a threshold or adding them as a special ignore node if the importance is below the threshold. When processing text, ignore nodes are allowed to match to any token. Experimentally, we found that a normalised activation threshold of 0.5 and an importance threshold of 0.75 worked well. The final important token is marked as a termination node, which represents a valid stopping point when processing text. It records the normalised activation of the activating node for this path in the trie. We repeat this process for all activating tokens in all the input prompts.

Text Processing: The resulting trie can be used to process text by beginning at the root of the trie and working backwards through the text prompt, checking if any consecutive sequence of prior tokens matches any path through the trie and reaches a termination node. We collate all valid matching paths and return the stored normalised activation on the longest matching path.

Visualisation: To visualize the trie, we create a condensed graph representation. We remove ignore nodes and termination nodes and create a layered graph by de-duplicating nodes by token value at each depth in the trie. We then color the activating nodes in the graph according to their normalised activation, with stronger activations corresponding to brighter red. Similarly, we color the rest of the token nodes in blue according to their importance. Additionally, we indicate nodes connected to a termination node in the full trie with a bold outline.

3 Results and Discussion

As the neuron graphs that are built by the algorithm can be directly used to process text and predict token activations, we can evaluate the degree to which they accurately capture the target neuron's behaviour by measuring the correspondence between the activations of the neuron and the predicted activations of the graph on some evaluation text. In our experiments we use a six-layer decoder-only Transformer model with SoLU activation, which may improve model interpretability by reducing polysemanticity Elhage et al. (2022a). The model is trained on the Pile (Gao et al., 2020), and we use data from Neuroscope (Nanda, 2022a), which provides token level activations for the top 20 prompts in the model's training set with the highest neuron activation on any token within the prompt, for all neurons in the model.

For each neuron, we take these top 20 dataset examples and randomly split them in half to form a train and test set, and give the training examples to N2G to create a neuron graph. We then take the test examples and normalise the token activations as described above. We apply a threshold to the token activations, defining an activation above the threshold as a *firing* of the neuron, and an activation below the threshold as the neuron *not firing*. In these experiments we set the threshold to 0.5. We then process the test prompts with the neuron graph to produce predicted token firings. We can then measure the precision, recall, and $F1$ score of the graph's predictions compared to the ground truth firings.

Table 3 shows the average precision, recall, and $F1$ score of the neuron graphs for a random sample of 50 neurons from each layer of the model, stratified by neuron firing. In layer 0, the graphs on average capture the behaviour of the neurons well, with high recall and good precision on the tokens for which the real neuron fires, whilst maintaining near-perfect recall on the much larger number of tokens for which the neuron does not fire. Note that predicting token-level firings is in general a very imbalanced problem, as neurons typically fire on a small proportion of tokens in the input prompts.

However, as we progress to deeper layers of the model, the recall and precision of the graphs generally decrease. This corresponds to neurons in the later layers on average exhibiting more complex

	Firing Tokens			Non-Firing Tokens		
Layer	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
0	0.74	0.85	0.74	1.0	0.99	1.0
1	0.66	0.77	0.64	1.0	1.0	1.0
2	0.60	0.77	0.6	1.0	1.0	1.0
3	0.48	0.70	0.48	1.0	0.99	1.0
4	0.44	0.72	0.46	1.0	0.99	1.0
5	0.45	0.67	0.42	1.0	0.99	1.0

Table 1: Precision, recall and $F1$-score of the neuron graphs' token-level predictions of neuron firing compared to ground truth on held-out test data, for 50 random neurons from each layer of the model. Tokens on which the real neuron fired and tokens on which it didn't fire are evaluated separately as there are generally many more tokens on which a neuron didn't fire, making it trivially easy to get near-perfect scores by always predicting the neuron will not fire.

behaviour that is less completely captured in the training examples. Specifically, neurons in early layers tend to respond to a small number of specific tokens in specific, narrow contexts, whereas later layers often respond to more abstract concepts represented by a wider array of tokens in many different contexts, which was similarly observed by (Elhage et al., 2022a). Precision also drops as the graphs may over-generalise and fail to capture the nuances of the context which caused a neuron to activate on a given token.

For example, Figure 3 shows a comparison between a graph from Layer 0 and Layer 3. The graph from Layer 0 is typical for that layer - a small number of activating nodes that activate in simple contexts, often requiring just one of a small set of possible prior tokens to be present, and sometimes requiring no additional context at all. In contrast, the graph from Layer 3 exhibits a more complex structure, with longer and more intricate context that captures the more abstract concept of software licensing required for activation on the activating nodes.

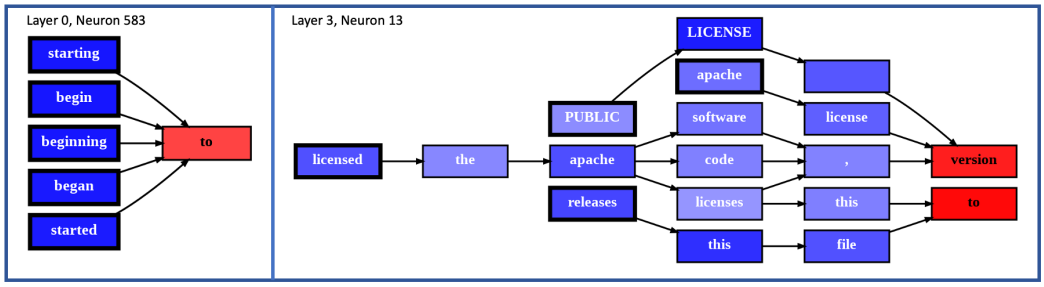

Figure 3: Left: a graph from Layer 0 of the model. Right: a graph from Layer 3 of the model.

4 Conclusions and Limitations

We introduced N2G, a method for automatically converting neurons in LLMs into interpretable graphs which can be visualized. The degree to which a graph captures the behaviour of a target neuron can be directly measured by comparing the output of the graph to the activations of the neuron, making this method a step towards scalable interpretability methods for LLMs. We find the neuron graphs capture neuron behaviour well for early layers of the model, but only partially capture the behaviour for later layers due to increasingly complex neurion behaviour, and this problem would likely become more prominent in larger models. Our approach primarily used SoLU (Elhage et al., 2022a) models to reduce polysemanticity. Although also applicable to models with typical activation functions, the resulting graphs may need to be more comprehensive due to more complex neuron behaviours. Our study focused on predicting neuron behaviour on the text that most activates it, excluding weaker activations. Future work could address these limitations by utilizing more training examples, better exploring the input space, and generalizing from exact token matches to matching abstract concepts, for example by using embeddings.

REFERENCES

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

Fazl Barez, Hosein Hasanbeig, and Alessandro Abate. System III: Learning with domain knowledge for safety constraints. In *NeurIPS ML Safety Workshop*, 2022. URL https://openreview.net/forum?id=85mcrDoWOAH.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. Identifying and controlling important neurons in neural machine translation. *CoRR*, abs/1811.01157, 2018. URL http://arxiv.org/abs/1811.01157.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*, 2021a.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda B. Viégas, and Martin Wattenberg. An interpretability illusion for BERT. *CoRR*, abs/2104.07143, 2021b. URL https://arxiv.org/abs/2104.07143.

Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. In *NIPS*, 2017.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. Knowledge neurons in pretrained transformers. *CoRR*, abs/2104.08696, 2021. URL https://arxiv.org/abs/2104.08696.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. What is one grain of sand in the desert? analyzing individual neurons in deep NLP models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):6309–6317, July 2019. doi: 10.1609/aaai.v33i01.33016309. URL https://doi.org/10.1609/aaai.v33i01.33016309.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL http://arxiv.org/abs/1810.04805. cite arxiv:1810.04805Comment: 13 pages.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. Analyzing individual neurons in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4865–4880, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.395. URL https://aclanthology.org/2020.emnlp-main.395.

Nelson Elhage, Tristan Hume, Catherine Olsson, Neel Nanda, Tom Henighan, Scott Johnston, Sheer ElShowk, Nicholas Joseph, Nova DasSarma, Ben Mann, Danny Hernandez, Amanda Askell, Kamal Ndousse, Jones, , Dawn Drain, Anna Chen, Yuntao Bai, Deep Ganguli, Liane Lovitt, Zac Hatfield-Dodds, Jackson Kernion, Tom Conerly, Shauna Kravec, Stanislav Fort, Saurav Kadavath, Josh Jacobson, Eli Tran-Johnson, Jared Kaplan, Jack Clark, Tom Brown, Sam McCandlish, Dario Amodei, and Christopher Olah. Softmax linear units. *Transformer Circuits Thread*, 2022a. https://transformer-circuits.pub/2022/solu/index.html.

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

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*, 2020.

Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 6(3):e30, 2021.

Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*, 2021.

Shusen Liu, Zhimin Li, Tao Li, Vivek Srikumar, Valerio Pascucci, and Peer-Timo Bremer. Nlize: A perturbation-driven visual interrogation tool for analyzing and interpreting natural language inference models. *IEEE transactions on visualization and computer graphics*, 25(1):651–660, 2018.

Edward Ma. Nlp augmentation. https://github.com/makcedward/nlpaug, 2019.

Jesse Mu and Jacob Andreas. Compositional explanations of neurons. *CoRR*, abs/2006.14032, 2020. URL https://arxiv.org/abs/2006.14032.

Neel Nanda. Neuroscope, 2022a. URL https://neuroscope.io/index.html.

Neel Nanda. Transformerlens, 2022b. URL https://github.com/neelnanda-io/TransformerLens.

Chris Olah. Mechanistic interpretability, variables, and the importance of interpretable bases. *Transformer Circuits Thread(June 27). http://www. transformer-circuits. pub/2022/mech-interp-essay/index. html*, 2022.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. doi: 10.23915/distill.00024.001. https://distill.pub/2020/circuits/zoom-in.

Nina Poerner, Benjamin Roth, and Hinrich Schütze. Interpretable textual neuron representations for NLP. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 325–327, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5437. URL https://aclanthology.org/W18-5437.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Ludwig Schubert, Chelsea Voss, Nick Cammarata, Gabriel Goh, and Chris Olah. High-low frequency detectors. *Distill*, 6(1):e00024–005, 2021.

Esther Seyffarth, Younes Samih, Laura Kallmeyer, and Hassan Sajjad. Implicit representations of event properties within contextual language models: Searching for "causativity neurons". In *Proceedings of the 14th International Conference on Computational Semantics (IWCS)*, pp. 110–120, Groningen, The Netherlands (online), June 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.iwcs-1.11.

Lucas Torroba Hennigen, Adina Williams, and Ryan Cotterell. Intrinsic probing through dimension selection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 197–216, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.15. URL https://aclanthology.org/2020.emnlp-main.15.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. Investigating Gender Bias in Language Models Using Causal Mediation Analysis. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12388–12401. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/92650b2e92217715fe312e6fa7b90d82-Paper.pdf.

John Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. Similarity analysis of contextual word representation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4638–4655, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.422. URL https://aclanthology.org/2020.acl-main.422.

A Appendix

A.1 Related Work

Prior work in neuron analysis has identified the presence of neurons correlated with specific concepts (Radford et al., 2017). For instance, Dalvi et al. (2019) explored neurons which specialised in linguistic and non-linguistic concepts in large language models, and Seyffarth et al. (2021) evaluated neurons which handle concepts such as causation in language. The existence of similar concepts embedded within models can also be found across different architectures. Wu et al. (2020) and Schubert et al. (2021) examined neuron distributions across models and found that different architectures have similar localised representations of information, even when Durrani et al. (2020) used a combination of neuron analysis and visualization techniques to compare transformer and recurrent models, finding that the transformer produces fewer neurons but exhibits stronger dynamics.

There are various methods of identifying concept neurons (Geva et al., 2020). Bau et al. (2018) proposed a method of identifying important neurons across models by analyzing correlations between neurons from different models. In contrast, Dai et al. (2021) developed a method to identify concept neurons in transformer feed-forward networks by computing the contribution of each neuron to the knowledge prediction. In contrast, we focus on identifying neurons using highly activating dataset examples. Mu & Andreas (2020) demonstrated how the co-variance of neuron activations on a dataset can be used to distinguish neurons that are related to a particular concept. Torroba Hennigen et al. (2020) also used neuron activations to train a probe which automatically evaluates language models for neurons correlated to linguistic concepts.

One limitation of using highly activating dataset examples is that the accurate identification of concepts correlated with a neuron is limited by the dataset itself. A neuron may represent several concepts, and Bolukbasi et al. (2021b) emphasise the importance of conducting interpretability research on varied datasets, in order to avoid the "interpretability illusion", in which neurons that show consistent patterns of activation in one dataset activate on different concepts in another. Poerner et al. (2018) also showed the limitations of datasets in concept neuron identification. They demonstrated that generating synthetic language inputs that maximise the activations of a neuron surpasses naive search on a corpus.

A.2 Neuron Graphs

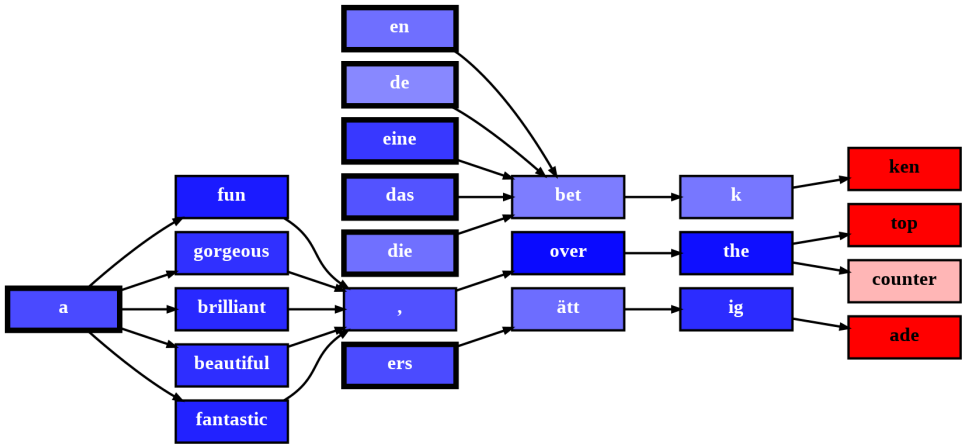

Figure 4: A neuron graph exhibiting polysemanticity, with three disconnected subgraphs each responding to a phrase in a different language.

In this section we explore some more interesting or characteristic behaviours of the neuron graphs. Polysemanticity, the phenomenon where a neuron exhibits multiple unrelated behaviours, is one of the current major challenges of neuron interpretability (Elhage et al., 2022b). Interestingly, when present, polysemanticity often shows up clearly in the neuron graphs as distinct, disconnected sub-

graphs. For example, in Figure 4, there are three separate subgraphs corresponding to three clearly distinct behaviours. The top subgraph responds to a phrase in Dutch - variations on *de betrokken*, where not all tokens in *betrokken* were important enough to include in the graph. The middle subgraph responds to a phrase in English - vartions on *a fun, over the top*. The bottom subgraph responds to a phrase in Swedish - *kollegers berättigade*, with unimportant tokens not included. This natural separation of behaviours into separate subgraphs could potentially make it easier to interpret polysemantic neurons, but more experimentation would be needed to develop and test this further.

N2G may work particularly well for neurons that respond to structured text such as code. For example, Figure 5 shows a graph for a neuron that responds to the syntax for *if* statements containing closing brackets. The current implementation of N2G appears to work better for this *syntactic* neurons than for more complex *semantic* neurons, suggesting it might be a useful tool for interpreting code models, and also highlighting the room for future development to enhance the ability to capture complex semantic behaviour as well.

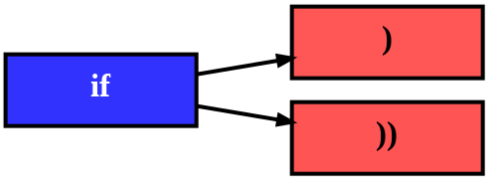

Figure 5: A neuron graph representing the syntax for *if* statements with closing brackets.

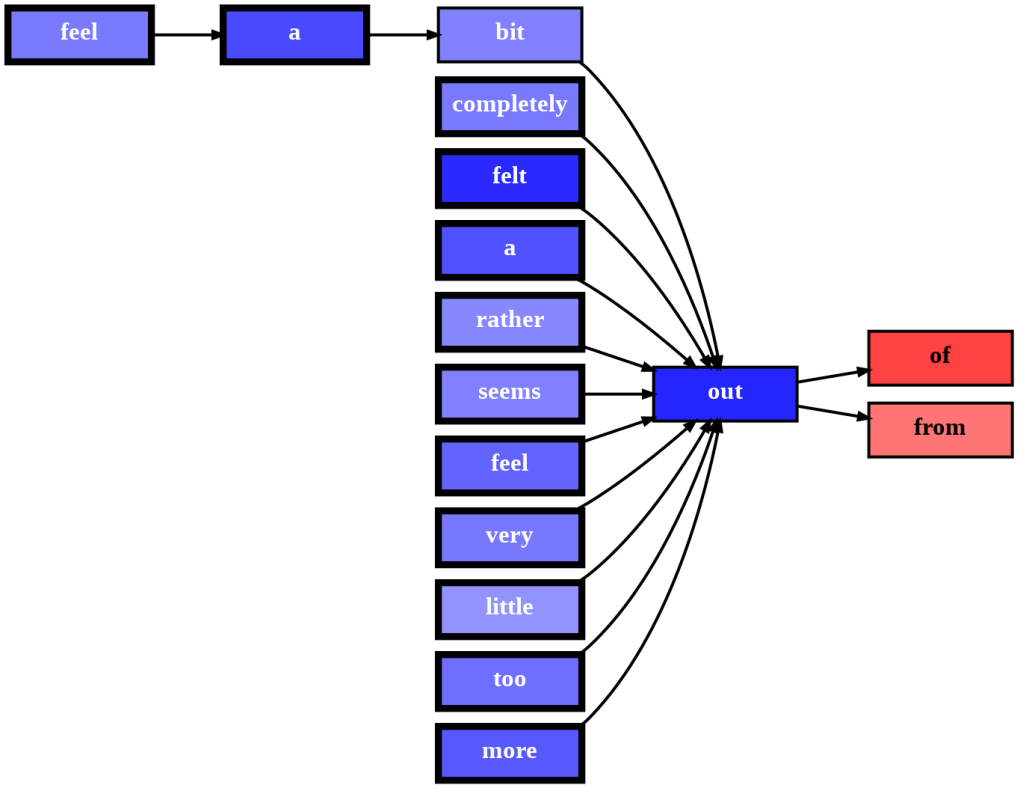

Figure 6: A neuron graph with the core token *out* that creates a bottleneck in the graph, as it must always be present for neuron activation despite not being an activating token itself.