NeuroX Library for Neuron Analysis of Deep NLP Models

Fahim Dalvi
Qatar Computing Research Institute, HBKU
faimaduddin@hbku.edu.qa

Hassan Sajjad*
Faculty of Computer Science
Dalhousie University
hsajjad@dal.ca

Nadir Durrani
Qatar Computing Research Institute, HBKU
ndurrani@hbku.edu.qa

Abstract

Neuron analysis provides insights into how knowledge is structured in representations and discovers the role of neurons in the network. In addition to developing an understanding of our models, neuron analysis enables various applications such as debiasing, domain adaptation and architectural search. We present *NeuroX*, a comprehensive open-source toolkit to conduct neuron analysis of natural language processing models. It implements various interpretation methods under a unified API, and provides a framework for data processing and evaluation, thus making it easier for researchers and practitioners to perform neuron analysis. The Python toolkit is available at https://www.github.com/fdalvi/NeuroX.[1]

1 Introduction

Interpretation of deep learning models is an essential attribute of trustworthy AI. Researchers have proposed a diverse set of methods to interpret models and answered questions such as: what linguistic phenomena are learned within representations, and what are the salient neurons in the network. For instance, a large body of work analyzed the concepts learned within representations of pre-trained models (Belinkov et al., 2017; Liu et al., 2019; Tenney et al., 2019; Rogers et al., 2020) and showed the presence of core-linguistic knowledge in various parts of the network. Several researchers have carried out this interpretation at a fine-grained level of neurons e.g. Durrani et al. (2020); Torroba Hennigen et al. (2020); Antverg and Belinkov (2021) highlighted salient neurons w.r.t any linguistic property in the model and Lundberg and Lee (2017); Dhamdhere et al. (2018); Janizek et al. (2020) identified a set of neurons responsible for a given prediction. At a broader level, these methods can be categorized into representation analysis, neuron analysis and feature attribution methods respectively. Sajjad et al. (2022) provides a comprehensive survey of these methods.

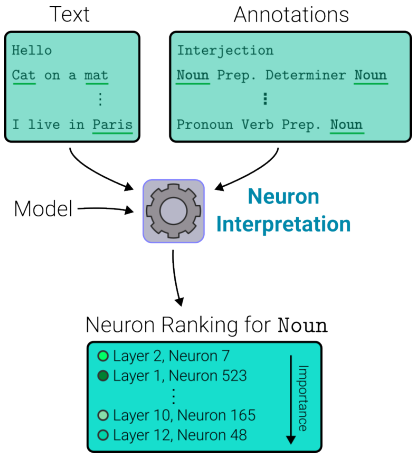


Figure 1: Simplified overview of Neuron Interpretation. Given an annotated text corpus, neuron interpretation methods aim to provide a ranking of neurons in a model w.r.t to their importance to one or more annotated properties (for e.g. "Noun" in this instance)

A number of toolkits have been proposed to facilitate the interpretation of deep learning models. For instance, diagNNose (Jumelet, 2020) provides representation analysis and attribution methods. LIT (Tenney et al., 2020) can be used to visualize attention and counterfactual explanations using feature attribution methods. Captum (Kokhlikyan et al., 2020) integrates a large set of attribution methods under a consistent API. All these tools facilitate the interpretability of models. However, due to the diverse ways of interpreting models, they do not cover all sets of methods. Specifically, most of the toolkits do not cover *neuron interpretation* methods that discover and rank neurons with respect to a concept.

Neuron interpretation analyzes and gives insight into how knowledge is structured within a representation. It discovers neurons with respect to a

*The work was done while the author was at QCRI.

[1]Demo Video available here: https://youtu.be/mLhs2YMx4u8

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 3: System Demonstrations, pages 226–234

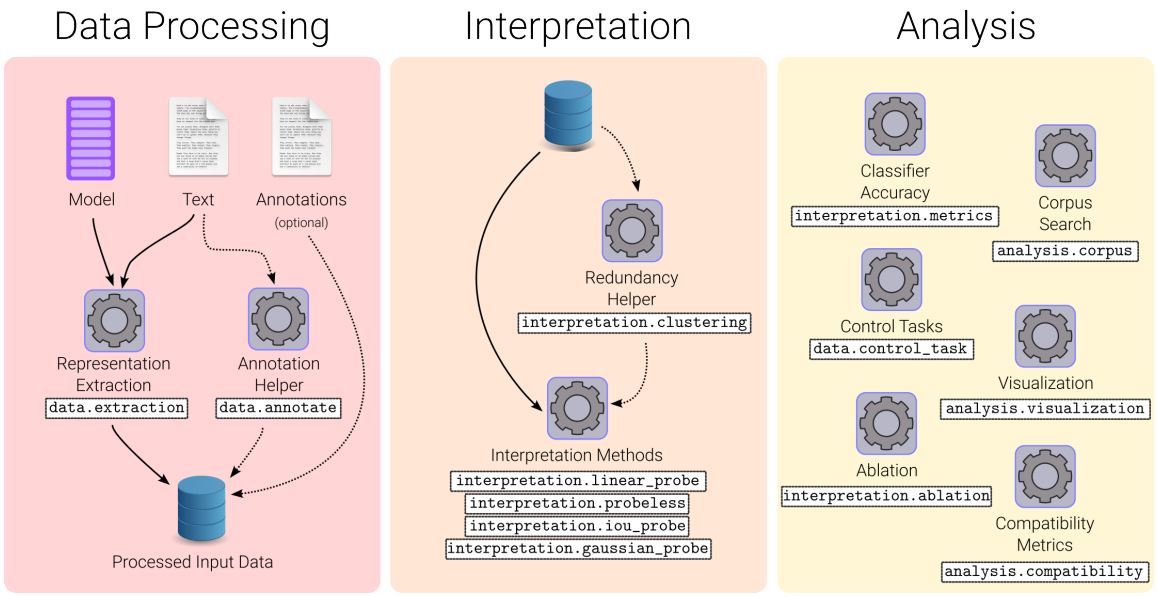


Figure 2: Overall design and architecture of the NeuroX toolkit, with references to their corresponding Python modules in the white boxes.

concept and provides a fine-grained interpretation of deep models. Figure 1 provides a simplified high-level overview of how neuron interpretation methods operate. Given a model, some text and annotations, these methods output a ranking of neurons with respect to their importance to one or more annotated concepts. The ability to interpret neurons enables applications such as debiasing models, controlling predictions of the models on the fly (Bau et al., 2019; Suau et al., 2020), neural architectural search (Dalvi et al., 2020) and domain adaptation (Gu et al., 2021). To make neuron interpretation more accessible, we propose *NeuroX*, an open-source Python toolkit to facilitate neuron interpretation of deep natural language processing (NLP) models.

NeuroX consists of three major components: i) data processing, ii) interpretation and iii) analysis. The data processing implements various ways to generate and upload data for analysis, extract activations and save them efficiently. The interpretation module implements six interpretation methods belonging to two different classes of methods. The analysis module brings together qualitative and quantitative methods to evaluate and visualize the discovered neurons. Figure 2 shows these components and how they interact with each other. We describe them in detail in the following sections. The toolkit itself is compatible with HuggingFace's transformers (Wolf et al., 2020) API and supports all transformer-based models.

To the best of our knowledge, *NeuroX* is the first toolkit that enables the interpretation of deep NLP models at the neuron level. It serves as a backbone to rapidly test new interpretation techniques using a unified framework and enables comparison and consistent evaluation of these techniques. The toolkit is easy to install and run:

```
pip install neurox
```

with detailed documentation is available at `https://neurox.qcri.org/docs`, including tutorials that showcase various capabilities of the toolkit to quickly get started with neuron interpretation.

2 Data Processing

The `data` module is responsible for preparing all the inputs for the Interpretation and Analysis modules, as well as filtering the datasets to probe and interpret specific phenomena. As shown in Figure 2, the required inputs to the toolkit are: i) a model and ii) a text corpus annotated towards the property of interest (e.g. data annotated towards toxic word spans in the hate-speech-detection task). The interpretation module can then extract a neuron ranking, highlighting the saliency of the neurons in the model that capture this phenomenon. If annotations are not available, an annotation helper module is made available in the toolkit that can annotate tokens based on arbitrary phenomena e.g. suffixation, lexical properties, or using pre-existing vocabularies. Below we describe the various components of the `data` module in detail.

Tokenizer	Input Sentence	Tokenized Sentence
bert-base-cased	"A good-looking house"	"[CLS] A good - looking house [SEP]"
gpt2	"A good-looking house"	"A Ä good - looking Ä house"
bert-base-cased	"Mauritians"	"[CLS] ma ##uri ##tian ##s [SEP]"
gpt2	"Mauritians"	"M aur it ians"
flaubert/flaubert_base_cased	"sport qu' on"	"<s> sport</w> qu</w> '</w> on</w> </s>"
flaubert/flaubert_base_cased	"sport qu'"	"<s> sport</w> qu'</w> </s>"

Table 1: Tokenizers from different models tokenize the same input very differently, sometimes adding special characters at the first subword, or prefixing all subwords except the first subword etc. Sometimes the same model tokenizes the same word (qu') differently depending on the context.

2.1 Representation Extraction

Central to any neuron interpretation method are the neuron activations themselves, i.e. the magnitude of a neuron for any given input. While modern frameworks such as PyTorch and Tensorflow facilitate the extraction of intermediate neuron values for specific models via hooks, it is non-trivial to enable this generically, as the code to extract activations from specific network components (e.g. layers) is highly dependent on the underlying model implementation. *NeuroX* implements generic extractors for specific popular frameworks and provides a highly-configurable PyTorch-based extractor.

Framework Specific Extractors An example of a framework specific extractor is one for HuggingFace's *transformers* models. The *transformers* library exposes the intermediate output at each layer, which can then be used to access each neuron's (layer output) activation for any given input.

Generic Extractors Apart from framework specific extractors, the toolkit offers a generic extractor for PyTorch model, which runs as a two step process. In the first step, the toolkit maps out the architecture of the given model, and provides the user a `json` file that contains all the components of the model. The user can then choose exactly which of the components they need the activations for, which are then saved in the second step.

Segmentation and De-Segmentation A unique problem to text and NLP models is that of tokenization. For instance, every *transformers* model has an associated *tokenizer*, that breaks the tokens in an input sentence into subwords depending on the model's vocabulary. The same input can be tokenized differently by each model. To get a neuron's activation for a given input token regardless of tokenization, *NeuroX* runs a detokenization procedure to combine the activation values on subwords into a single activation value. Table 1 shows examples of how a sentence (and sometimes a word) can be tokenized differently depending on the underlying tokenizer and context. The toolkit also offers the user a choice on how the activation values across subwords should be combined such as `first or last subword` or `average across subwords`.

2.2 Annotation Helper

While annotations are available for some linguistic properties, labeled data sets may not always be available. To carry out an interpretation in such a scenario, *NeuroX* offers a helper module that can label the data with a positive or negative label per token. `data.annotate.annotate_data` can annotate each token positively in three different ways:

1. **Preset Vocabulary:** The token exists in the given vocabulary.
2. **Regular expression:** The token matches with the given regular expression. For example, the expression `^\d+$` annotates all tokens that are composed of digits as positive samples.
3. **Python function:** A function that returns a binary True/False for a given token. Arbitrary computation can be done inside this function. For instance, `lambda token: token.endswith("ing")` annotates all tokens ending with *-ing* positively.

3 Interpretation Module

The central module in the *NeuroX* toolkit is the `interpretation` module, which provides implementations of several neuron and representation analysis methods. Table 2 shows a list of methods that are currently implemented in the toolkit, along with details of what each method's implementation supports.

The method implementations follow a consistent API to make it easy for the user to switch

Interpretation Method	Description	Supports Representation Analysis	Requires Training	Supports multi-class analysis
`Linear Probe`	Class of probing methods that use a linear classifier for neuron analysis. Specifically, the implementation provides probes introduced by • Radford et al. (2019) (Classifier with L1 regularization) • Lakretz et al. (2019) (Classifier with L2 regularization) • Dalvi et al. (2019a) (Classifier with Elastic Net regularization)	Yes	Yes	Yes
`Probeless`	A corpus-based neuron search method that obtains neuron rankings based on an accumulative strategy, introduced by Antverg and Belinkov (2021)	No	No	Yes
`IoU Probe`	A mask-based method introduced by Mu and Andreas (2020) that computes Intersection over Union between tokens representing a specific concept and tokens that have high activation values for specific neurons	No	No	No
`Gaussian Probe`	A multivariate Gaussian based classifier introduced by Torroba Hennigen et al. (2020) that can probe for neurons whose activation values follow a gaussian distribution.	Yes	Yes	Yes
`Mean Select`	A corpus-based neuron ranking method introduced by Fan et al. (2023) that derives neuron importances by looking at activation values across contexts where a concept appears.	No	No	Yes

Table 2: An overview of the neuron interpretation methods currently implemented in the *NeuroX* toolkit.

between them. Specifically, each method at least implements the following functions:

- `method.train_probe`: This function takes in the pre-processed data (extracted activations, prepared labels etc) as described in section 2, and returns back a probe that can be used to perform neuron analysis. Some methods do not require any training, in which case this function just stores the input for future use.
- `method.evaluate_probe`: This function takes an evaluation set and returns the performance of the probe on the given set. The evaluation set itself can be a control task, and the output score can be computed using several pre-implemented metrics. Section 4 discusses the various evaluation metrics in detail.
- `method.get_neuron_ordering`: This function returns an ordering/ranking of all the neurons being analyzed with respect to their importance to the task at hand. For instance, if the probe was trained to analyze *Nouns*, this function will return a sorted list of neurons (by importance) that activate for Nouns in the given dataset.

The interpretation methods themselves may be able to probe multiple properties at the same time (multi-class probing), or only a single concept (binary probing). Additionally, some interpretation methods can also perform representation-level analysis, i.e. probe an entire layer rather than individual neurons.

Redundancy Analysis: Dalvi et al. (2020) have shown that large neural networks learn knowledge redundantly where multiple neurons are optimized to activate on the same input. This is encouraged by the optimization choices such as dropouts which explicitly force neurons to learn in the absence of other neurons. In order to facilitate the analysis of redundant neurons, the toolkit provides a clustering based non-redundant neuron extraction method. Running the neurons through `interpretation.clustering.extract_independent_neurons` first before performing any probing can reduce the overall search space of neurons, and lead to better findings and analyses.

4 Analysis and Evaluation

The `analysis` module provides implementations of various evaluation and analysis methods. Some of these methods provide quantitative results like

accuracy scores, while others allow users to perform qualitative analysis on neurons.

4.1 Classifier Accuracy

Classifier accuracy reciprocates the probing framework (Belinkov et al., 2017; Hupkes et al., 2018). Once a neuron ranking is obtained, a classifier is trained towards the task of interest (the intrinsic concept for which the probe was originally trained) with the selected neurons. The delta between oracle performance (accuracy using all the neurons) and the accuracy of the classifier using the selected neurons measures the efficacy of the ranking.

Selectivity It is important to ensure that the probe is truly representing the concepts encoded within the learned representations and not memorizing them during classifier training. *NeuroX* enables control task selectivity, a measure proposed by Hewitt and Liang (2019) to mitigate memorization using the `data.control_task` module.

4.2 Ablation

An alternative approach used by (Dalvi et al., 2019a) is to ablate all but the selected neurons in the trained probe. The `interpretation.ablation` allows manipulating the input data by keeping/filtering specific neurons in the order of their importance, allowing users to measure the drop in performance with selected neurons.

4.3 Mutual Information

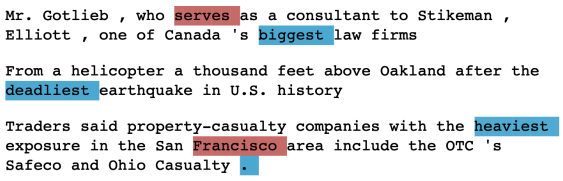
Information theoretic metrics such as mutual information have also been used to interpret representations of deep NLP models (Pimentel et al., 2020). Here, the goal is to measure the amount of information a representation provides about a linguistic concept. It is computed by calculating the mutual information between a subset of neurons and linguistic concepts.

4.4 Compatibility Metrics

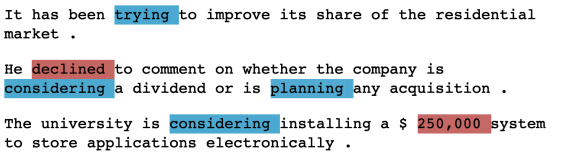
Another set of evaluation metrics recently proposed by Fan et al. (2023) carries out a pair-wise comparison of the discovered neurons across methods. While this strategy does not provide a direct evaluation of a neuron interpretation method, it provides an insight into how *compatible* a method is with the other available methods. *NeuroX* implements two compatibility metrics in the `analysis.compatibility` module: i) `Average Overlap` (which shows how aligned a method is with others) and ii) `NeuronVote` (which shows how well-endorsed the ranking of a method is by other methods).

4.5 Qualitative Evaluation

Visualizations have been used effectively to gain qualitative insights on analyzing neural networks (Karpathy et al., 2015; Kádár et al., 2017). *NeuroX* provides a text visualization module (`analysis.visualization`) that displays the activations of neurons w.r.t. to a concept (e.g. Figure 3). The toolkit also allows corpus-based analysis in the `analysis.corpus` module by extracting the top n words in a corpus that activate a neuron. Examples are shown in Table 3.



(a) Superlative Adjective Neuron



(b) Gerund Verb Neuron

Figure 3: Visualizations (POS) – Superlative Adjective and Gerund Verb Neurons

5 Miscellaneous Functions

5.1 Scalability

Extracting, saving and working with neuron activations over large datasets and models can be very expensive, since each neuron's activation is saved for each token in the input corpus. To enable both disk- and runtime-savings, *NeuroX* provides a low precision mode where all the activations are saved using 16-bit precision instead of the default 32/64-bit precision. This results in considerable storage/memory savings and also improves training/inference performance depending on the method and underlying hardware. The precision can be controlled by supplying the `dtype=float16` option to the extraction/interpretation methods.

5.2 Disk Formats for Representations

The toolkit offers flexibility to the user over the format used to save the neuron activations. Specifi-

Neuron	concept	Model	Top-5 words
Layer 9: 624	VBD	RoBERTa	supplied, deposited, supervised, paled, summoned
Layer 2: 750	VBG	RoBERTa	exciting, turning, seeing, owning, bonuses
Layer 0: 249	VBG	BERT	requiring, eliminating, creates, citing, happening
Layer 1: 585	VBZ	XLNet	achieves, drops, installments, steps, lapses, refunds
Layer 2: 254	CD	RoBERTa	23, 28, 7.567, 56, 43
Layer 5: 618	CD	BERT	360, 370, 712, 14.24, 550
Layer 1: 557	LOC	XLNet	Minneapolis, Polonnaruwa, Mwangura, Anuradhapura, Kobe
Layer 5: 343	ORG	RoBERTa	DIA, Horobets, Al-Anbar, IBRD, GSPC
Layer 10: 61	PER	RoBERTa	Grassley, Cornwall, Dalai, Bernanke, Mr.Yushchenko
Layer 6: 132	PER	BERT	Nick, Manie, Troy, Sam, Leith
Layer 2: 343	YOC	BERT	1897, 1918, 1901, 1920, Alam

Table 3: Ranked list of words for some individual neurons, VBD: Past-tense verb, VBG: Gerund Verb, VBZ: Third person singular, CD: Numbers, LOC: Location, ORG: Organization, PER: Person, YOC: Year of the century

cally, it offers readers and writers for a text-based format (`json`) and a binary format (`hdf5`). The binary format provides faster saving/loading performance, speeding up experiments with a large number of neurons or a large amount of text. On the other hand, the text-based format is considerably easier to debug.

6 Related Work

A number of toolkits have been made available to carry out the analysis and interpretation of neural network models. The What-If tool (Wexler et al., 2019) inspects machine learning models and provides users an insight into the trained model based on the predictions. Seq2Seq-Vis (Strobelt et al., 2018) enables the user to trace back the prediction decisions to the input in neural machine translation models. Captum (Kokhlikyan et al., 2020) provides generic implementations of a number of gradient and perturbation-based attribution algorithms. LIT (Tenney et al., 2020) implements various methods of counterfactual explanations, attribution methods and visualization of attention. diagNNose (Jumelet, 2020) integrates representation analysis methods and attribution methods and finally, iModelsX (Singh and Gao, 2023) aims to provide natural explanations for datasets, which can provide insights into the models that are trained with these datasets. While these tools cover a number of interpretation methods, none of them facilitate neuron-level interpretation of NLP models. The LM-Debugger toolkit (Geva et al., 2022) is an interactive debugger for transformer LMs, which provides a fine-grained interpretation and a powerful framework for intervening in LM behavior.

Ecco (Alammar, 2021) is a visualization based library that implements saliency methods and additionally enables visualization of neurons of the network. Similar to Ecco, the *NeuroX* toolkit enables visualization of neurons of the network. In addition, we implement a wide range of neuron interpretation methods that can be accessed using a uniform API and provide various analysis and evaluation methods. Our toolkit empowers researchers to focus on specific parts of the neuron interpretation research such as interpretation, comparison or evaluation without worrying about setting up the rest of the pipeline like data processing, embedding extraction, integration with various pre-trained models, and evaluation of the method. NeuroX powers other interpretation analysis frameworks such as ConceptX (Alam et al., 2022) and NxPlain (Dalvi et al., 2023).

The previous version of NeuroX (Dalvi et al., 2019b) only supported a specific machine translation library and one neuron interpretation method (Dalvi et al., 2019a) as a GUI app. The current Python toolkit is a redesigned version with a unified architecture. It includes multiple features like a data processing module, numerous neuron interpretation and evaluation methods, and seamless integration with popular toolkits such as HuggingFace's *transformers*.

7 Conclusion and Future Work

We presented *NeuroX*, an open-source toolkit to carry out neuron-level interpretation of representations learned in deep NLP models. It maintains implementations of several neuron analysis methods under a consistent API, and provides implementations for preparing the data, analyzing neurons and evaluating the methods. In the future, *NeuroX* plans to expand its extraction module to other frameworks like FairSeq and OpenNMT-py. In ad-

dition, we plan to integrate attribution based neuron saliency methods to add another class of interpretation methods to the toolkit.

8 Acknowledgements

We are grateful to all NeuroX contributors and users who have provided bug reports to improve the toolkit. Specifically, we would like to thank Ahmed Abdelali for testing and reporting bugs with the extraction implementations, David Arps for scalability improvements, Yimin Fan for implementing several interpretation methods and Yifan Zhang for working on detailed documentation and tutorials.

9 Ethical Considerations

The NeuroX toolkit provides a post hoc interpretation of pre-trained models. The toolkit makes a contribution towards improving the transparency of deep models and may discover biases present in these models. We do not foresee any direct ethical issues with respect to the developed toolkit. In terms of the neuron interpretation methods, the majority of them are based on the correlation between neurons and the input. One potential issue with such an interpretation is its faithfulness with respect to the knowledge used by the model in making predictions. However, this is not a limitation of the toolkit but a limitation of the research methods in general.

References

Firoj Alam, Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Abdul Rafae Khan, and Jia Xu. 2022. Conceptx: A framework for latent concept analysis.

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.

Omer Antverg and Yonatan Belinkov. 2021. On the pitfalls of analyzing individual neurons in language models. In *International Conference on Learning Representations*.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do Neural Machine Translation Models Learn about Morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, Vancouver. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, D. Anthony Bau, and James Glass. 2019a. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI, Oral presentation)*.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Tamim Jaban, Mus'ab Husaini, and Ummar Abbas. 2023. NxPlain: A web-based tool for discovery of latent concepts. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 75–83, Dubrovnik, Croatia. Association for Computational Linguistics.

Fahim Dalvi, Avery Nortonsmith, D Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, and James Glass. 2019b. Neurox: A toolkit for analyzing individual neurons in neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Fahim Dalvi, Hassan Sajjad, Nadir Durrani, and Yonatan Belinkov. 2020. Analyzing redundancy in pretrained transformer models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP-2020)*, Online.

Kedar Dhamdhere, Mukund Sundararajan, and Qiqi Yan. 2018. How important is a neuron? *CoRR*, abs/1805.12233.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4865–4880, Online. Association for Computational Linguistics.

Yimin Fan, Fahim Dalvi, Nadir Durrani, and Hassan Sajjad. 2023. Evaluating neuron interpretation methods of nlp models.

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

Shuhao Gu, Yang Feng, and Wanying Xie. 2021. Pruning-then-expanding model for domain adaptation of neural machine translation.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. 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 2733–2743, Hong Kong, China.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure.

Joseph D. Janizek, Pascal Sturmfels, and Su-In Lee. 2020. Explaining explanations: Axiomatic feature interactions for deep networks.

Jaap Jumelet. 2020. diagNNose: A library for neural activation analysis. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 342–350, Online. Association for Computational Linguistics.

Ákos Kádár, Grzegorz Chrupała, and Afra Alishahi. 2017. Representation of linguistic form and function in recurrent neural networks. *Computational Linguistics*, 43(4):761–780.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

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.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc.

Jesse Mu and Jacob Andreas. 2020. Compositional explanations of neurons. *CoRR*, abs/2006.14032.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020. Information-theoretic probing for linguistic structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Hassan Sajjad, Nadir Durrani, and Fahim Dalvi. 2022. Neuron-level Interpretation of Deep NLP Models: A Survey. *Transactions of the Association for Computational Linguistics*.

Chandan Singh and Jianfeng Gao. 2023. Emb-GAM: an interpretable and efficient predictor using pre-trained language models.

Hendrik Strobelt, Sebastian Gehrmann, Michael Behrisch, Adam Perer, Hanspeter Pfister, and Alexander Rush. 2018. Debugging sequence-to-sequence models with Seq2Seq-vis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 368–370, Brussels, Belgium. Association for Computational Linguistics.

Xavier Suau, Luca Zappella, and Nicholas Apostoloff. 2020. Finding experts in transformer models. *CoRR*, abs/2005.07647.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

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.

Lucas Torroba Hennigen, Adina Williams, and Ryan Cotterell. 2020. Intrinsic probing through dimension selection. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 197–216, Online. Association for Computational Linguistics.

James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, 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. pages 38–45. Association for Computational Linguistics.