Ecco: An Open Source Library for the Explainability of Transformer Language Models

J Alammar
Arpeggio Research
ar.pegg.io
jay.alammar@pegg.io

Abstract

Our understanding of *why* Transformer-based NLP models have been achieving their recent success lags behind our ability to continue scaling these models. To increase the transparency of Transformer-based language models, we present Ecco – an open-source[1] library for the explainability of Transformer-based NLP models. Ecco provides a set of tools to capture, analyze, visualize, and interactively explore inner mechanics of these models. This includes (1) gradient-based feature attribution for natural language generation (2) hidden states and their evolution between model layers (3) convenient access and examination tools for neuron activations in the under-explored Feed-Forward Neural Network sublayer of Transformer layers. (4) convenient examination of activation vectors via canonical correlation analysis (CCA), non-negative matrix factorization (NMF), and probing classifiers. We find that syntactic information can be retrieved from BERT's FFNN representations in levels comparable to those in hidden state representations. More curiously, we find that the model builds up syntactic information in its hidden states *even* when intermediate FFNNs indicate diminished levels of syntactic information. Ecco is available at https://www.eccox.io/.[2]

1 Introduction

The Transformer architecture (Vaswani et al., 2017) has been powering many recent advances in NLP. A breakdown of this architecture is provided by Alammar (2018) and will help understand this paper's details. Pre-trained language models based on the architecture (Liu et al., 2018; Devlin et al., 2018; Radford et al., 2018, 2019; Liu et al., 2019; Brown et al., 2020) continue to push the envelope in various tasks in NLP and, more recently, in computer vision (Dosovitskiy et al., 2020). Our understanding of why these models work so well, however, still lags behind these developments.

Ecco provides tools and interactive explorable explanations[3] aiding the examination and intuition of:

- **Input saliency** methods that score input tokens importance to generating a token are discussed in section 2.
- **Hidden state evolution** across the layers of the model and what it may tell us about each layer's role. This is discussed in section 3.
- **Neuron activations** and how individual and groups of model neurons spike in response to inputs and to produce outputs. This is discussed in section 4.

[1]The code is available at https://github.com/jalammar/ecco

[2]Video demo available at https://youtu.be/bcEysXmR09c

[3]http://worrydream.com/ExplorableExplanations/

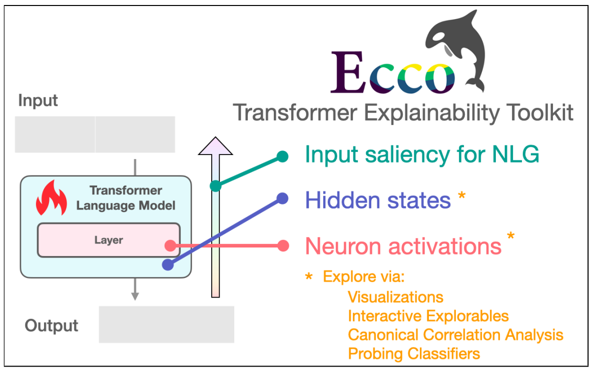


Figure 1: **A set of tools to make the inner workings of Transformer language models more transparent.** By introducing tools that analyze and visualize input saliency (for natural language generation), hidden states, and neuron activations, we aim to enable researchers to build more intuition about Transformer language models.

- **Non-negative matrix factorization of neuron activations** to uncover underlying patterns of neuron firings, revealing firing patterns of linguistic properties of input tokens. This is discussed in subsection 4.2.

Ecco creates rich, interactive interfaces directly inside Jupyter notebooks (Ragan-Kelley et al., 2014) running on pre-trained models from the Hugging Face transformers library (Wolf et al., 2020). Currently it supports GPT2 (Radford et al., 2018), BERT (Devlin et al., 2018), and RoBERTa (Liu et al., 2019). Support for more models and explainability methods is under development and open for community contribution.

2 Input Saliency

When a computer vision model classifies a picture as containing a husky, an input saliency map (Figure 2) can tell us whether the classification was made due to the visual properties of the animal itself or because of the snow in the background (Ribeiro et al., 2016). This is a method of attribution explaining the relationship between a model's output and inputs – helping us detect errors and biases to better understand the system's behavior.

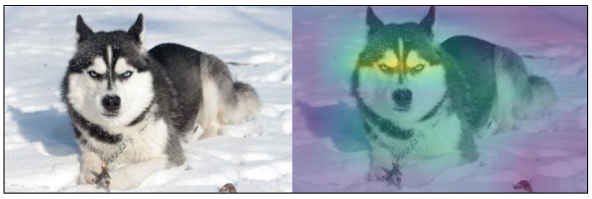
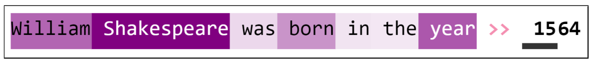


Figure 2: Input saliency map attribute a model's prediction to input pixels.

Multiple methods exist for assigning feature importance scores to the inputs of an NLP model (Li et al., 2015; Arrieta et al., 2020). Instead of assigning scores to pixels, in the NLP domain these methods assign scores to input tokens. The literature is most often concerned with this application for classification tasks rather than natural language generation. Ecco enables generating output tokens and then interactively exploring the saliency values for each output token.

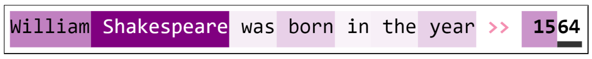
2.1 Saliency View

In Figure 3, we see an experiment to probe the world knowledge of GPT2-XL. We ask the model to output William Shakespeare's date of birth. The model is correctly able to produce the date (1564, but broken into two tokens: *15* and *64*, because the model's vocabulary does not include *1564* as a single token). By hovering on each token, Ecco imposes each input's saliency value as a background color. The darker the color, the more that input token is attributed responsibility for generating this output token.



William Shakespeare was born in the year >> 1564

(a) Input saliency for the first output token, *15* (shown by hovering over *15*).



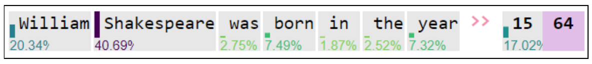
William Shakespeare was born in the year >> 1564

(b) Input saliency for the second output token, *64* (shown by hovering over *64*).

Figure 3: **GPT2-XL is able to tell the birth date of William Shakespeare.** It expresses it in two tokens: *15* and *64*. Ecco shows the input saliency of each of these tokens using Gradient X Inputs. The darker the background color of the token is, the higher its saliency value.

2.2 Detailed Saliency View

Ecco also provides a detailed view to see the attribution values in more precision. Figure 4 demonstrates this interactive interface which displays the normalized attribution value as a percentage and bar next to each token.



William	Shakespeare	was	born	in	the	year	>>	15	64
20.34%	40.69%	2.75%	7.49%	1.87%	2.52%	7.32%		17.02%	

Figure 4: Ecco's detailed input saliency view for the token *64* (shown by hovering over *64*).

About Gradient-Based Saliency Ecco calculates feature importance based on Gradients X Inputs (Denil et al., 2015; Shrikumar et al., 2017) – a gradient-based saliency method shown by Atanasova et al. (2020) to perform well across various datasets for text classification in Transformer models.

Gradients X Inputs can be calculated using the following formula:

$$\|\nabla_{X_i} f_c(X_{1:n}) X_i\|_2$$

Where X_i is the embedding vector of the input token at timestep i, and $\nabla_{X_i} f_c(X_{1:n})$ is the back-propagated gradient of the score of the selected token. The resulting vector is then aggregated into a score via calculating the L2 norm as this was

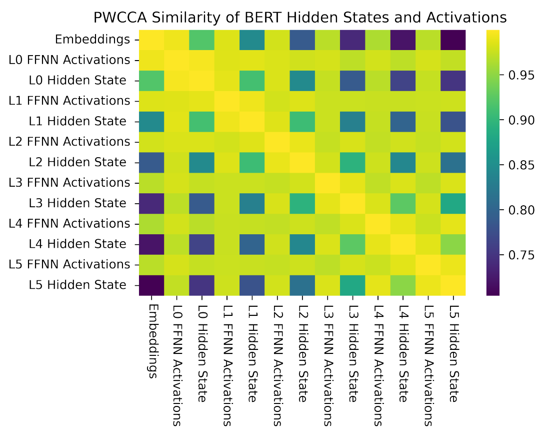
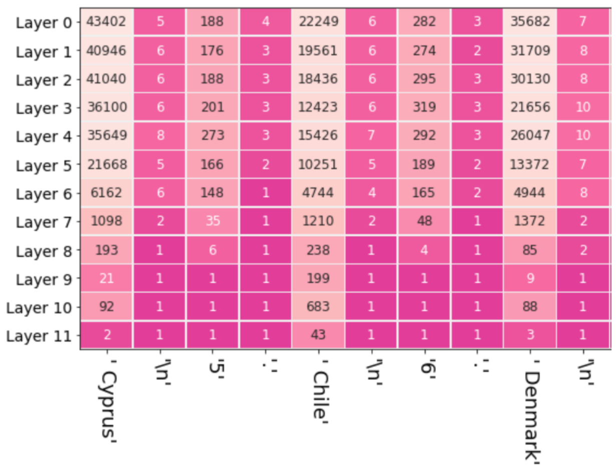


Figure 6: **Similarity of hidden states and FFNN activations in a distilled BERT model.** Ecco enables capturing neuron activations and comparing activation space similarity using Projection Weighted Canonical Correlation Analysis (PWCCA).



	' Cyprus'	'\n'	'5'	'.'	' Chile'	'\n'	'6'	'.'	' Denmark'	'\n'
Layer 0	43402	5	188	4	22249	6	282	3	35682	7
Layer 1	40946	6	176	3	19561	6	274	2	31709	8
Layer 2	41040	6	188	3	18436	6	295	3	30130	8
Layer 3	36100	6	201	3	12423	6	319	3	21656	10
Layer 4	35649	8	273	3	15426	7	292	3	26047	10
Layer 5	21668	5	166	2	10251	5	189	2	13372	7
Layer 6	6162	6	148	1	4744	4	165	2	4944	8
Layer 7	1098	2	35	1	1210	2	48	1	1372	2
Layer 8	193	1	6	1	238	1	4	1	85	2
Layer 9	21	1	1	1	199	1	1	1	9	1
Layer 10	92	1	1	1	683	1	1	1	88	1
Layer 11	2	1	1	1	43	1	1	1	3	1

Figure 7: **Evolution of the rankings of a list of countries across the 12 layers of GPT2-XL**. The prediction represented here is generated using GPT2-XL, on the input sequence `"The countries of the European Union are:\n1. Austria\n2. Belgium\n3. Bulgaria\n4"`. Output decoding strategy used is top50 sampling.

empirically shown by Atanasova et al. (2020) to perform better than other methods.

3 Hidden States Examination

Another method to glean information about the inner workings of a language is by examining the hidden states produced by every Transformer block. Ecco provides multiple methods to examine the hidden states and to visualize how they evolve across the layers of the model.

3.1 Canonical Correlation Analysis (CCA)

Recent work has used Canonical Correlation Analysis (Hotelling, 1992) to examine language model internal representations. For example, Voita et al. (2019) used hidden state to analyze the flow of information inside Transformers and how the informational content of hidden states compares across tasks. Singh et al. (2019) examined internal representations of multilingual BERT. Wu et al. (2020) compared the internal representations of multiple NLP models. More specifically, these works used recently developed methods like SVCCA (Raghu et al., 2017), PWCCA (Morcos et al., 2018) and CKA (Kornblith et al., 2019).

Ecco bundles these methods (`cca()`, `svcca()`, `pwcca()`, and `cka()`) to allow convenient similarity comparison of language model representations. This includes hidden state representations, yet also extends to neuron activations (Ecco pays special attention to the neurons after the largest dense FFNN layer as can be seen in Section 4). Figure 6 shows a comparison of the hidden states and FFNN neuron activations as the model processes textual input. All three CCA methods take two activation vectors (be they hidden states or neuron activations) and assign a similarity score from zero (no correlation) to one (the two inputs are linear transformations of each other).

3.2 Ranking of Output Token Across Layers

Nostalgebraist (2020) presents compelling visual treatments showcasing the evolution of token rankings, logit scores, and softmax probabilities for the evolving hidden state through the various layers of the model. The author does this by projecting the hidden state into the output vocabulary using the language model head (which is typically used only for the output of the final layer).

Ecco enables creating such plots as can be seen in Figure 7. More examples showcasing this method can be found in(Alammar, 2021).

3.3 Comparing Token Rankings

Ecco also allows asking questions about which of two tokens the model chooses to output for a specific position. This includes questions of subject-verb agreement like those posed by Linzen et al. (2016). In that task, we want to analyze the model's

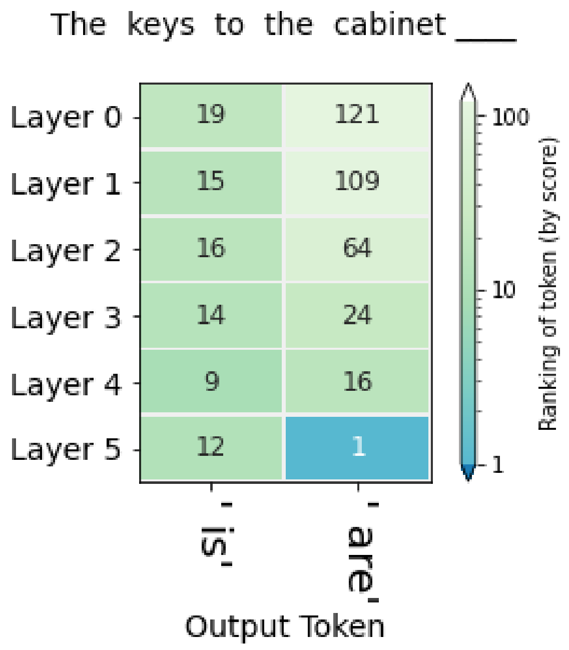


Figure 8: **Rankings, across model layers, of which token should go in the blank** DistillGPT-2 is prompted by the prompt shown at the top, while limited to two output tokens shown on the bottom.

capacity to encode syntactic number (whether the subject we're addressing is singular or plural) and syntactic subjecthood (which subject in the sentence we're addressing). Put simply, fill-in the blank. The only acceptable answers are 1) *is* 2) *are*:

The key**s** to the cabinet ______

Using Ecco, we can present this sentence to DistilGPT-2, and visualize the rankings of *is* and *are* using `ecco.rankings_watch()`, which creates Figure 8. The first column shows the rankings of the token *is* as the completion of the sentence, and the second column shows those for the token *are* for that same position. The model ultimately ranks *are* as the more probable answer, but the figure raises the question of why five layers fail to rank *are* higher than *is*, and only the final layer sets the record straight.

4 Neuron Activations

The Feed-Forward Neural Network (FFNN) sublayer is one of the two major components inside a Transformer block (in addition to self-attention). It often makes up two-thirds of a Transformer block's parameters, thus providing a significant portion of the model's representational capacity. Previous work (Karpathy et al., 2015; Strobelt et al., 2017; Poerner et al., 2018; Radford et al., 2017; Olah et al., 2017, 2018; Bau et al., 2018; Dalvi et al., 2019; Rethmeier et al., 2020) has examined neuron firings inside deep neural networks in both the NLP and computer vision domains. Ecco makes it easier to examine neuron activations by collecting them and providing tools to analyze them and reduce their dimensionality to extract underlying patterns.

4.1 Probing classifiers

Probing classifiers (Veldhoen et al., 2016; Adi et al., 2016; Conneau et al., 2018) are the most commonly used method for associating NLP model components with linguistic properties (Belinkov and Glass, 2019). Ecco currently supports linear probes with control tasks (Hewitt and Liang, 2019). Section 5 is a case study on using this method to probe FFNN representations for part-of-speech information.

4.2 Uncovering underlying patterns with NMF

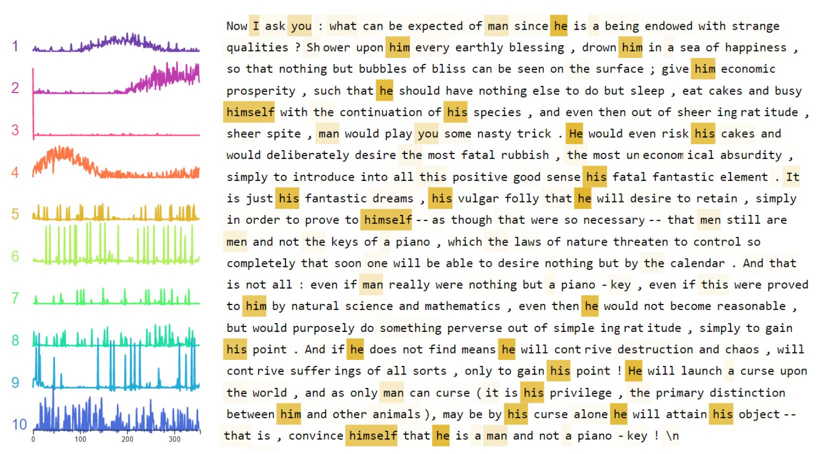
By first capturing the activations of the neurons in FFNN layers of the model and then decomposing them into a more manageable number of factors through NMF, we can shed light on how various neuron groups respond to input tokens.

Figure 9 shows intuitively interpretable firing patterns extracted from raw firings through NMF. This example, showcasing ten factors applied to the activations of layer #0 in response to a text passage, helps us identify neurons that respond to syntactic and semantic properties of the input text. The factor highlighted in this screenshot, factor 5, seems to correlate with pronouns.

This interface can compress a lot of data that showcase the excitement levels of factors (and, by extension, groups of neurons). The sparklines (Tufte, 2006) on the left give a snapshot of the excitement level of each factor across the entire sequence. Interacting with the sparklines (by hovering with a mouse or tapping) displays the activation of the factor on the tokens in the sequence on the right.

4.3 About Matrix Factorization of Neuron Activity

Figure 10 explains the intuition behind dimensionality reduction using NMF. This method can reveal underlying behavior common to groups of neurons. It can be used to analyze the entire network, a single layer, or groups of layers.



Now I ask you : what can be expected of man since he is a being endowed with strange
qualities ? Sh ower upon him every earthly blessing , drown him in a sea of happiness ,
so that nothing but bubbles of bliss can be seen on the surface ; give him economic
prosperity , such that he should have nothing else to do but sleep , eat cakes and busy
himself with the continuation of his species , and even then out of sheer ing rat itude ,
sheer spite , man would play you some nasty trick . He would even risk his cakes and
would deliberately desire the most fatal rubbish , the most un econom ical absurdity ,
simply to introduce into all this positive good sense his fatal fantastic element . It
is just his fantastic dreams , his vulgar folly that he will desire to retain , simply
in order to prove to himself -- as though that were so necessary -- that men still are
men and not the keys of a piano , which the laws of nature threaten to control so
completely that soon one will be able to desire nothing but by the calendar . And that
is not all : even if man really were nothing but a piano - key , even if this were proved
to him by natural science and mathematics , even then he would not become reasonable ,
but would purposely do something perverse out of simple ing rat itude , simply to gain
his point . And if he does not find means he will cont rive destruction and chaos , will
cont rive suffer ings of all sorts , only to gain his point ! He will launch a curse upon
the world , and as only man can curse (it is his privilege , the primary distinction
between him and other animals), may be by his curse alone he will attain his object --
that is , convince himself that he is a man and not a piano - key ! \n

Figure 9: **Individual factor view: Activation pattern in response to pronouns** Ten Factors extracted from the activations the neurons in Layer 0 in response to a passage from *Notes from Underground*. Hovering on the line graphs isolates the tokens of a single factor and imposes the magnitude of the factor's activation on the tokens as a background color. The darker the color the higher the activation magnitude. In addition to the pronouns factor highlighted in the figure, we can see factors that focus on specific regions of the text (beginning , middle , and end). This indicates neurons that are sensitive to positional encodings. View this interface online at (Alammar, 2020).

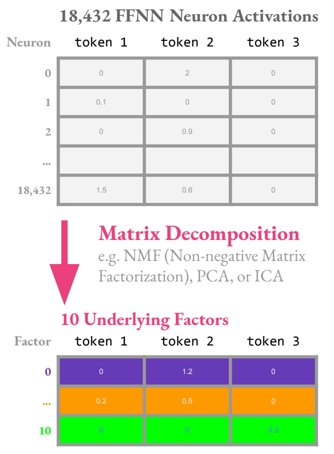


Figure 10: **Decomposition of activations matrix using Non-negative Matrix Factorization.** NMF reveals underlying patterns of neuron activations inside one layer, a collection of layers, or the entire model.

.

5 Case study: Probing FFNN neuron activations for PoS information

In this section, we use Ecco to examine the representations of BERT's Feed-Forward Neural Network using probing classifiers. Our work is most similar to Durrani et al. (2020). There has been plenty of work on probing BERT focused on the hidden states, but none to our knowledge that trained probes to extract token information from the FFNN representation.

5.1 Method

We first forward-pass the entire dataset through BERT. We capture all the hidden states of all the model's layers as well as the neuron activations of the FFNN sublayers (namely, the widest layer composed of 3072 neurons after the GELU activation). We then train external linear classifiers to predict the PoS of the tokens in the dataset and then report the accuracy on the test set. Because probes have been criticized as memorizing the inputs, we report selectivity scores (Hewitt and Liang, 2019) next to each accuracy score. Selectivity is metric that is calculated by generating a control task where each token is assigned a random part-of-speech tag. A separate probe is then trained on this control set. The difference in accuracy between the actual dataset and the control dataset is the selectivity score. The higher selectivity is, the more we can say that the probe really extracted part-of-speech data from the representation of the model as opposed to simply memorizing the training set.

5.2 Experimental Setup

We use the Universal Dependencies version 2 part-of-speech dataset in English. We extract 10,000 tokens and split them into a 67% and 33% train/test sets. We train linear probes for 50 epochs using the Adam optimizer. We run experiments with learning rates (0.1, 0.001, 1e-5) and report those of the best achieving learning rate (0.001). We run five trials and report their average results. For every trial, we train a probe for each permutation of 1) model layer 2) hidden state vs. FFNN activations 3) actual labels vs. random controls to calculate selectivity scores.

5.3 Results

We report accuracy and selectivity scores in Table 1. We observe that FFNN neuron activations do encode PoS information at levels comparable to hidden states. We find intriguing the divergence of scores in layers 2 and 3 between FFNN activations (which drop slightly) and hidden states (which continue increasing). Future work can examine if this divergence points towards layers storing different

	FFNN Activations				Hidden States (context. embeds)			
layer id	accuracy		selectivity		accuracy		selectivity	
Embed	- - -		- - -		**87.6**	(±0.5)	**6.1**	(±1.1)
0	90.5	(±0.4)	9.1	(±0.7)	92.2	(±0.3)	13.6	(±0.8)
1	93.3	(±0.5)	14.8	(±0.8)	93.6	(±0.4)	17.9	(± 1)
2	87.3	(±0.5)	35.9	(±0.5)	**94.2**	(±0.3)	20.9	(±0.8)
3	**84.7**	(±0.5)	34.6	(±0.3)	**94.7**	(±0.3)	23.9	(±0.7)
4	94.2	(±0.3)	18.9	(±0.6)	94.9	(±0.2)	27.5	(±0.6)
5	94.6	(±0.3)	22.2	(±0.6)	94.8	(±0.3)	31.5	(±0.7)
6	93.6	(±0.5)	27.1	(±1.1)	94.1	(±0.4)	34.3	(±0.8)
7	92.8	(±0.6)	31.5	(±0.5)	93.7	(±0.6)	36.0	(±0.4)
8	91.9	(±0.6)	34.4	(±1.1)	92.4	(±0.7)	37.1	(±0.5)
9	90.5	(±0.4)	35.2	(±0.4)	91.6	(±0.6)	37.3	(±0.7)
10	88.8	(±0.5)	36.4	(±0.6)	90.6	(±0.5)	37.3	(±0.9)
11	87.9	(±0.5)	36.5	(±0.8)	89.0	(±0.7)	36.7	(±1.1)

Table 1: Probing BERT representations for Part-of-Speech information. **We can see that raw embeddings already have some PoS information encoded, but the low selectivity indicates this accuracy score is inflated. (Embed layer)** The model continues to build PoS information through the first half of the network, increasing in both accuracy and selectivity (Layers 0-5). FFNN representations are comparable to hidden states in the quantity of PoS information our probes can extract. It is interesting that **a layer can increase PoS information** despite its FFNN showing **lower accuracy** (layer 3). This could indicate that different FFNN sublayers encode different subsets of PoS information and the model is able to extract only the subset of information that layer specializes in.

subsets of PoS information which the model is able to collect and assemble as it builds up its internal representations across layers.

6 System Design

Ecco is implemented as a python library that provides a wrapper around a pre-trained language model. The wrapper collects the required data from the language model (e.g., neuron activations, hidden states) and makes the needed calculations (e.g., input saliency, NMF dimensionality reduction). The interactive visualizations are built using web technologies manipulated through D3.js (Bostock et al., 2012).

Ecco is built on top of open source libraries including Scikit-Learn (Pedregosa et al., 2011), Matplotlib (Hunter, 2007), NumPy (Walt et al., 2011), PyTorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020). Canonical Correlation Analysis is calculated using the code open-sourced[4] by the authors (Raghu et al., 2017; Morcos et al., 2018; Kornblith et al., 2019).

7 Limitations

Ecco's input saliency feature is currently only supported for GPT2-based models, while neuron activation collection and dimensionality reduction are supported for GPT2 in addition to BERT and RoBERTa.

We echo the sentiment of Leavitt and Morcos (2020) that visualization has a role in building intuitions, but that researchers are encouraged to use that as a starting point towards building testable and falsifiable hypotheses of model interpretability.

8 Conclusion

As language models proliferate, more tools are needed to aid debugging models, explain their behavior, and build intuitions about their inner-mechanics. Ecco is one such tool combining ease of use, visual interactive explorables, and multiple model explainability methods.

Ecco is open-source software[5] and contributions are welcome.

Acknowledgments

This work was improved thanks to feedback provided by Abdullah Almaatouq, Anfal Alatawi, Christopher Olah, Fahd Alhazmi, Hadeel Al-Negheimish, Hend Al-Khalifa, Isabelle Augenstein, Jasmijn Bastings, Najwa Alghamdi, Pepa Atanasova, and Sebastian Gehrmann.

[4]https://github.com/google/svcca

[5]https://github.com/jalammar/ecco

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2016. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *arXiv preprint arXiv:1608.04207*.

J Alammar. 2018. The illustrated transformer.

J Alammar. 2020. Interfaces for explaining transformer language models.

J Alammar. 2021. Finding the words to say: Hidden state visualizations for language models.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. 2020. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115.

Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. 2020. A diagnostic study of explainability techniques for text classification.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Identifying and controlling important neurons in neural machine translation.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Mike Bostock et al. 2012. D3. js-data-driven documents.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. *arXiv preprint arXiv:1805.01070*.

Fahim Dalvi, Avery Nortonsmith, Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, and James Glass. 2019. Neurox: A toolkit for analyzing individual neurons in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9851–9852.

Misha Denil, Alban Demiraj, and Nando de Freitas. 2015. Extraction of salient sentences from labelled documents.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. *arXiv preprint arXiv:2010.02695*.

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. Association for Computational Linguistics.

Harold Hotelling. 1992. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer.

John D Hunter. 2007. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR.

Matthew L. Leavitt and Ari Morcos. 2020. Towards falsifiable interpretability research.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2015. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

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

Ari Morcos, Maithra Raghu, and Samy Bengio. 2018. Insights on representational similarity in neural networks with canonical correlation. In S. Bengio,

H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 5732–5741. Curran Associates, Inc.

Nostalgebraist. 2020. interpreting gpt: the logit lens.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature visualization. *Distill*, 2(11):e7.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. 2018. The building blocks of interpretability. *Distill*, 3(3):e10.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Nina Poerner, Benjamin Roth, and Hinrich Schütze. 2018. Interpretable textual neuron representations for nlp. *arXiv preprint arXiv:1809.07291*.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

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

Min Ragan-Kelley, F Perez, B Granger, T Kluyver, P Ivanov, J Frederic, and M Bussonnier. 2014. The jupyter/ipython architecture: a unified view of computational research, from interactive exploration to communication and publication. *AGUFM*, 2014:H44D–07.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. 2017. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. 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 6076–6085. Curran Associates, Inc.

Nils Rethmeier, Vageesh Kumar Saxena, and Isabelle Augenstein. 2020. Tx-ray: Quantifying and explaining model-knowledge transfer in (un-) supervised nlp. In *Conference on Uncertainty in Artificial Intelligence*, pages 440–449. PMLR.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "why should i trust you?": Explaining the predictions of any classifier.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2017. Not just a black box: Learning important features through propagating activation differences.

Jasdeep Singh, Bryan McCann, Richard Socher, and Caiming Xiong. 2019. BERT is not an interlingua and the bias of tokenization. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 47–55, Hong Kong, China. Association for Computational Linguistics.

Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush. 2017. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks.

Edward R Tufte. 2006. *Beautiful evidence*. Graphis Pr.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sara Veldhoen, Dieuwke Hupkes, and Willem H Zuidema. 2016. Diagnostic classifiers revealing how neural networks process hierarchical structure. In *CoCo@ NIPS*.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. *arXiv preprint arXiv:1909.01380*.

Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. 2011. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi 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 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.

John Wu, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2020. Similarity analysis of contextual word representation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4638–4655, Online. Association for Computational Linguistics.