

Fine-grained Interpretation and Causation Analysis in Deep NLP Models



Hassan
Sajjad



Narine
Kokhlikyan



Fahim
Dalvi



Nadir
Durrani

Link to Slides and Video

<https://github.com/hsajjad/Interpretability-Tutorial-NAACL2021>

Motivation

- Deep neural models: state-of-the-art for many tasks



- Issue: opaqueness
- Interpretation is important
 - Better understanding
 - Increasing trust in AI systems
 - Assisting ethical decision making
 - ...

What is Interpretation?

What knowledge does a model **learn** and **use** to solve a problem?

Is this knowledge similar to what humans rely on (linguistic knowledge such as morphology, syntactic structure etc.) or is it completely unexpected and unintuitive

What is Interpretation?

What knowledge does a model **learn** and **use** to solve a problem?

Is this knowledge similar to what humans rely on (linguistic knowledge such as morphology, syntactic structure etc.) or is it completely unexpected and unintuitive

How is knowledge distributed across the model components?

What knowledge learned within the model is used for specific predictions?

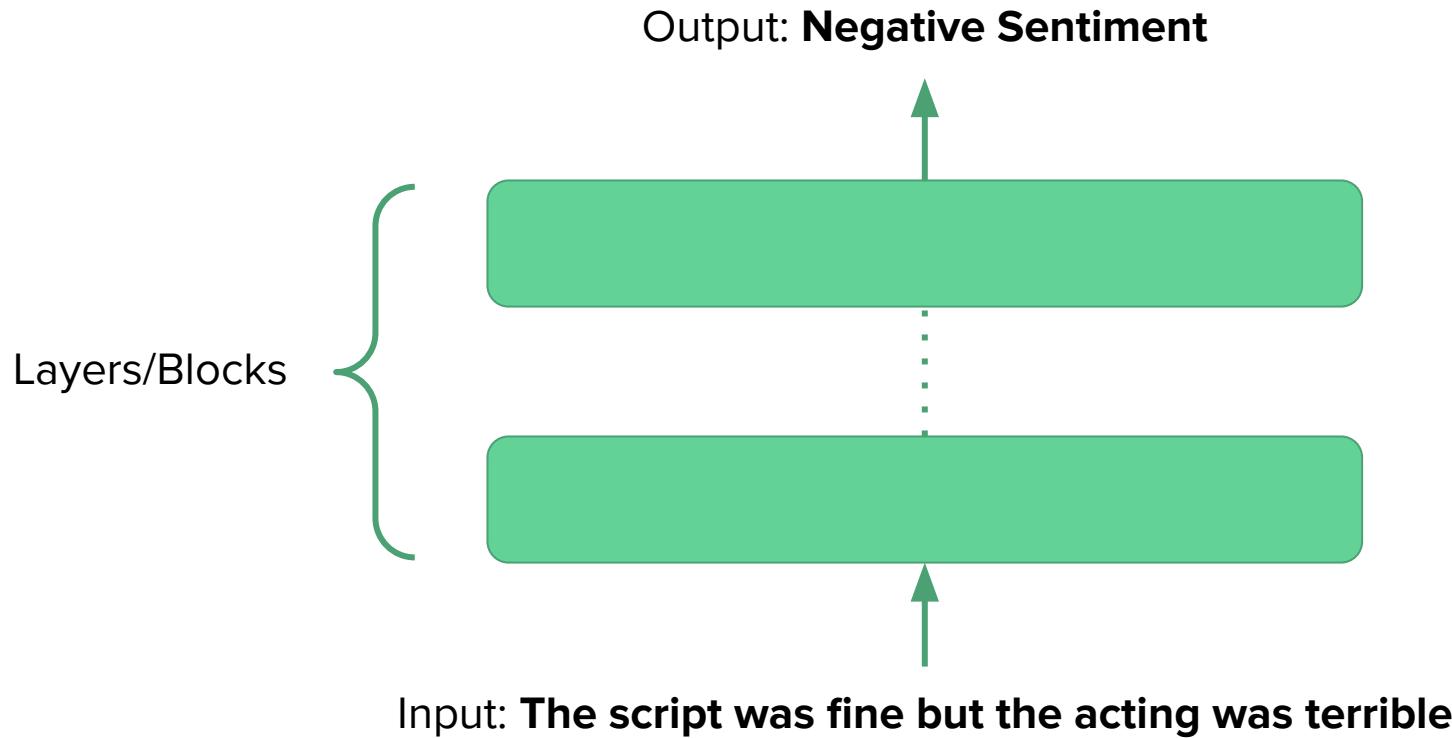
Does the inhibition of specific knowledge in the model change predictions?

How do different modeling and optimization choices impact the underlying knowledge?

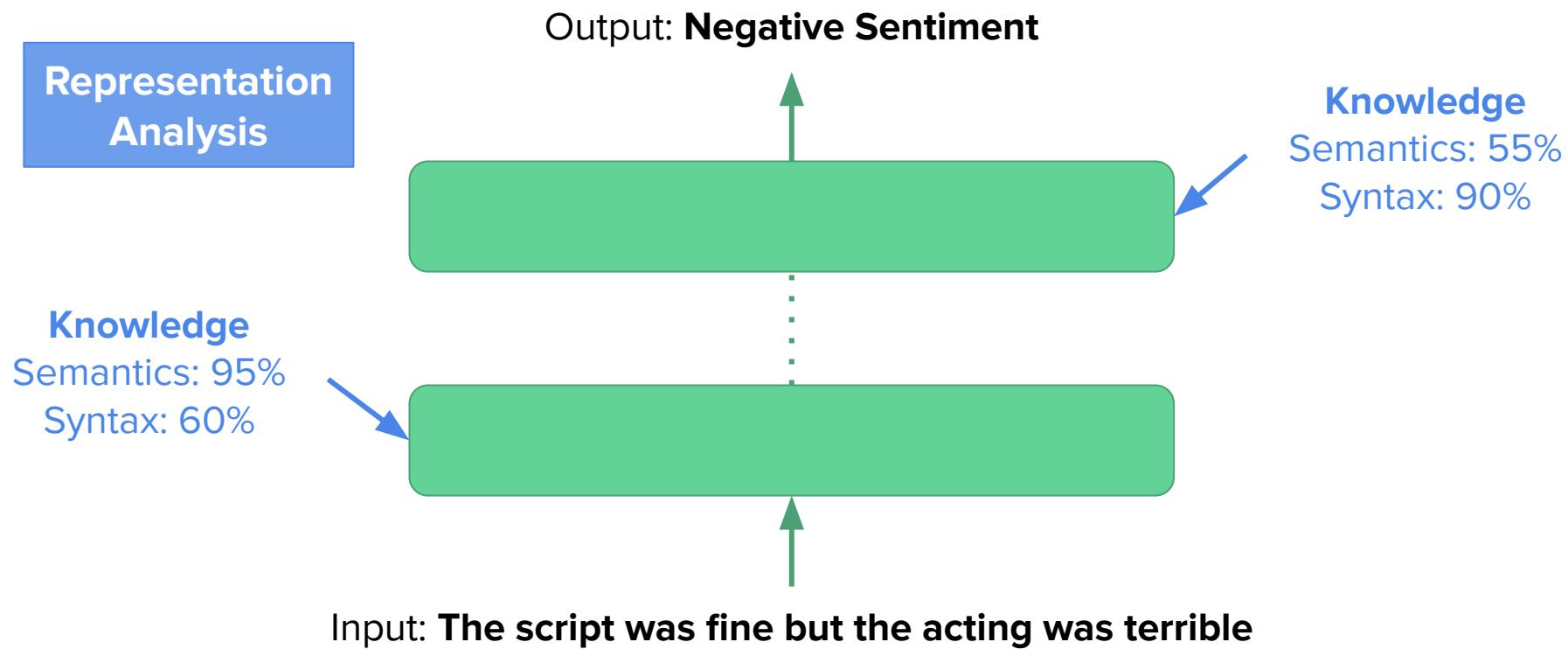
Previous Tutorials

- Interpretability and Analysis in Neural NLP (ACL 2020)
 - Representation analysis via probing classifiers
 - Behavioural analysis
 - Visualization and interactions
- Interpreting Predictions of NLP Models (EMNLP 2020)
 - What parts of input lead to prediction?
 - How certain global decision rules lead to a prediction?
 - What is the role of training examples in prediction?
- Explaining Machine Learning Predictions: State-of-the-art, Challenges, and Opportunities (NeurIPS 2020)
 - Feature Importances
 - Attribution-based methods
 - Counterfactual explanations
 - Collection of local explanations and counterfactual explanations

Previous Tutorials



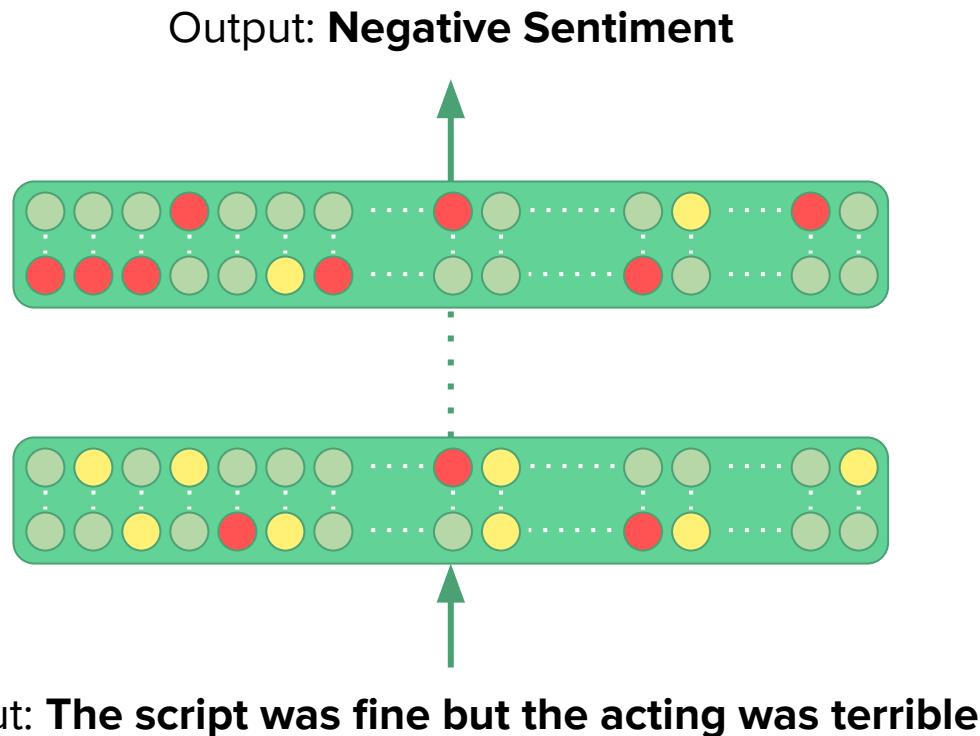
Previous Tutorials



This Tutorial

Neuron
Analysis

Knowledge
Semantics: ●
Syntax: ●



This Tutorial

Output: Negative Sentiment

N
Pr

Representation analysis shows
What is encoded in the representation?

Know
Sema
Syr

Neuron analysis shows
How the knowledge is encoded in the representation?

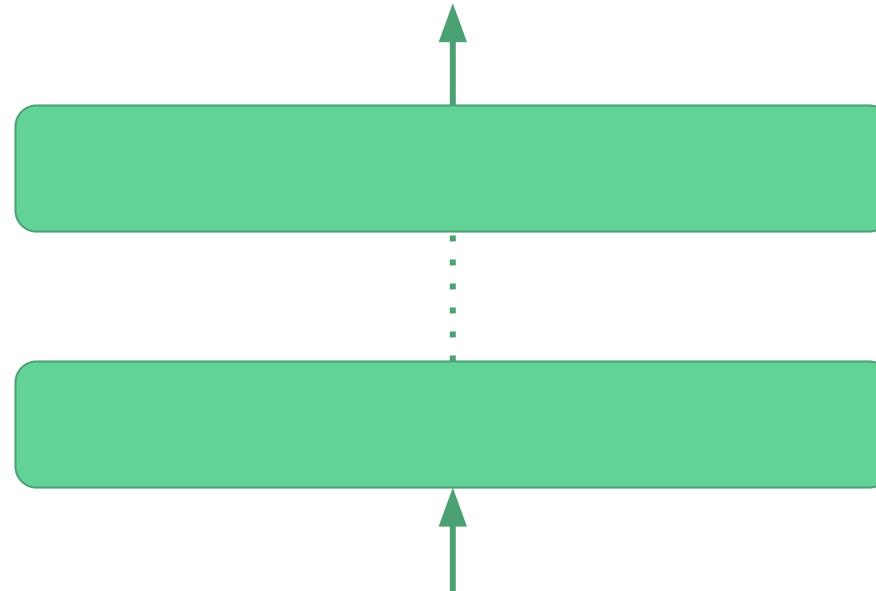
Fine-grained Analysis

Input: The script was fine but the acting was terrible

Previous Tutorials

**Input Feature
Importance**

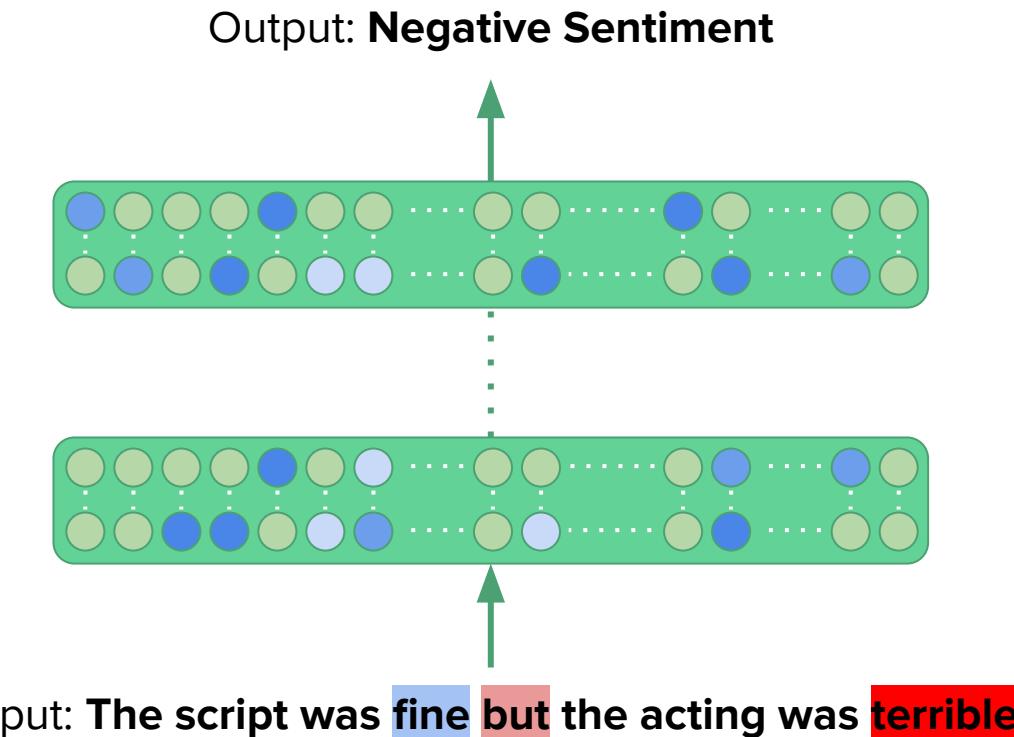
Output: Negative Sentiment



Input: The script was fine but the acting was terrible

This Tutorial

Neuron
Importance



This Tutorial

Output: Negative Sentiment

Neuron
Importance

Feature importance shows

What input features are important with respect to a particular prediction?

Neuron Importance shows

What neurons are important with respect to a particular prediction?

Causation Analysis

Input: The script was **fine** but the acting was **terrible**

This Tutorial

Formally, neuron-level analysis **unfolds the knowledge learned in the representation.** It answers questions such as:

- How the knowledge is encoded in the representation?
- How various concepts are learned among neurons?
- Which individual neurons or groups of neurons are critical for the model?
- Which neurons play a role in making a prediction?

Highlights

Neurons Learning Specific Concepts

- Individual neurons, referred here as a unit, learn specific words and phrases

Unit 108: **legal**, **law**, **legislative**

- Better **legal** protection for accident victims.
- These rights are guaranteed under **law**.
- This should be guaranteed by **law**.
- This **legislative** proposal is unusual.
- Animal feed must be safe for animal health.

Unit 711: **should**, **would**, **not**, **can**

- That **would not** be democratic.
- That **would** be cheap and it **would not** be right.
- This is **not** how it **should** be in a democracy.
- I hope that you **would not** want that!
- Europe **cannot** and must **not** tolerate this.

Switch Neuron

- A neuron learning **present and past verb tense** on the opposite spectrum of their activation values

7439th meeting , held on 11 May 2015 .

ISIL itself has published videos depicting people being subjected to a range of abhorrent punishments , including stoning , being pushed-off buildings , decapitation and crucifixion .

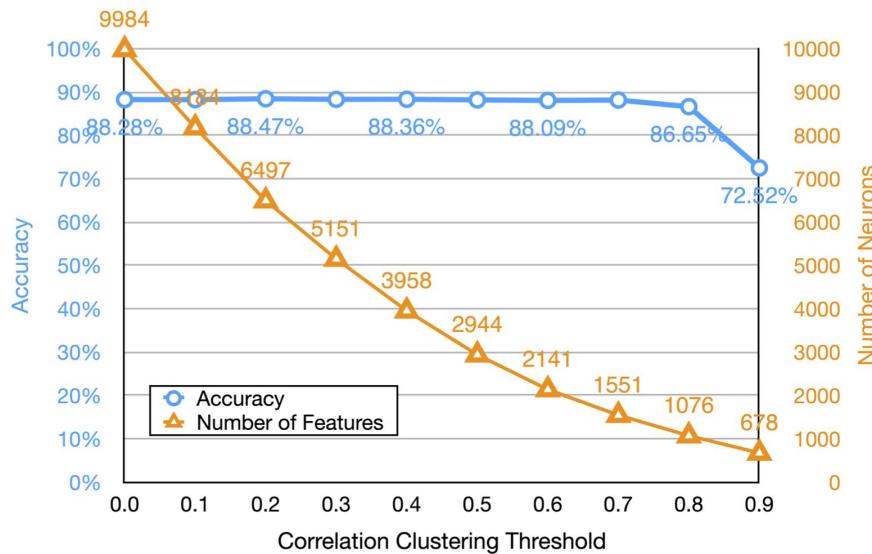
UNICEF disbursed emergency cash assistance to tens of thousands of displaced families in camps and UNHCR discontinued cash assistance to vulnerable families which had been internally displaced .

31 . Recognizes the important contribution of the African Peer Review Mechanism since its inception in improving governance and supporting socioeconomic development in African countries , and recalls in this regard the high-level panel discussion held on 21 October 2013 on Africa 's innovation in governance through 10 years of the African Peer Review Mechanism , organized during the sixty-eighth session of the General Assembly to commemorate the tenth anniversary of the Mechanism ;

Spreads between sovereign bonds in Germany and those in other countries were relatively unaffected by political and market uncertainties concerning Greece in late 2014 and early 2015 .

Redundancy Amongst Neurons

- A large number of neurons are redundant with respect to downstream tasks



Features from BERT	Downstream task accuracy
9984 features	88.28%
1551 features	88.09%

Spurious Heuristic Neurons

- Neurons learning inductive biases

Unit 39 (nobody in hypothesis)

hyp:nobody AND (NOT pre:hair) AND (NOT pre:RB) AND (NOT pre:'s)

IoU **0.465** w_{entail} **-0.117** w_{neutral} **-0.053** w_{contra} **0.047**

Pre Three women prepare a meal in a kitchen.

Orig Hyp The ladies are cooking.

Adv Hyp **Nobody but** the ladies are cooking.

True entail $\xrightarrow{\text{adv}}$ neutral Pred entail $\xrightarrow{\text{adv}}$ contra

Neuron Attribution

- Aggregated neuron attributions for each token in the 10-th layer of Bert sentiment classification model

[CLS] it is such a wonderful movie [SEP] [PAD] [PAD]

Legend: ■ Negative □ Neutral ■ Positive

Other Applications

Knowing the role and importance of neurons enable various applications beyond interpretation, such as:

- Model distillation
- Domain adaptation
- Model manipulation
- Architectural search
- Generalization

Road Map

Road Map

Fine-grained Analysis

Interprets how **various language or task specific concepts** are learned and represented within neurons in the network

Formally, we refer to this class of techniques as “Concept Analysis” in this tutorial

Causation Analysis

Highlights which neurons in the network were **used in making a particular prediction**

Formally, we refer to this class of techniques as “Attribution-based Methods” in this tutorial

Connecting the two worlds: concept-based interpretation of model’s prediction

Road Map

Concept Analysis

- Methods
- Evaluation techniques
 - Quantitative evaluation
 - Qualitative evaluation
- Datasets
- Findings
- Practical

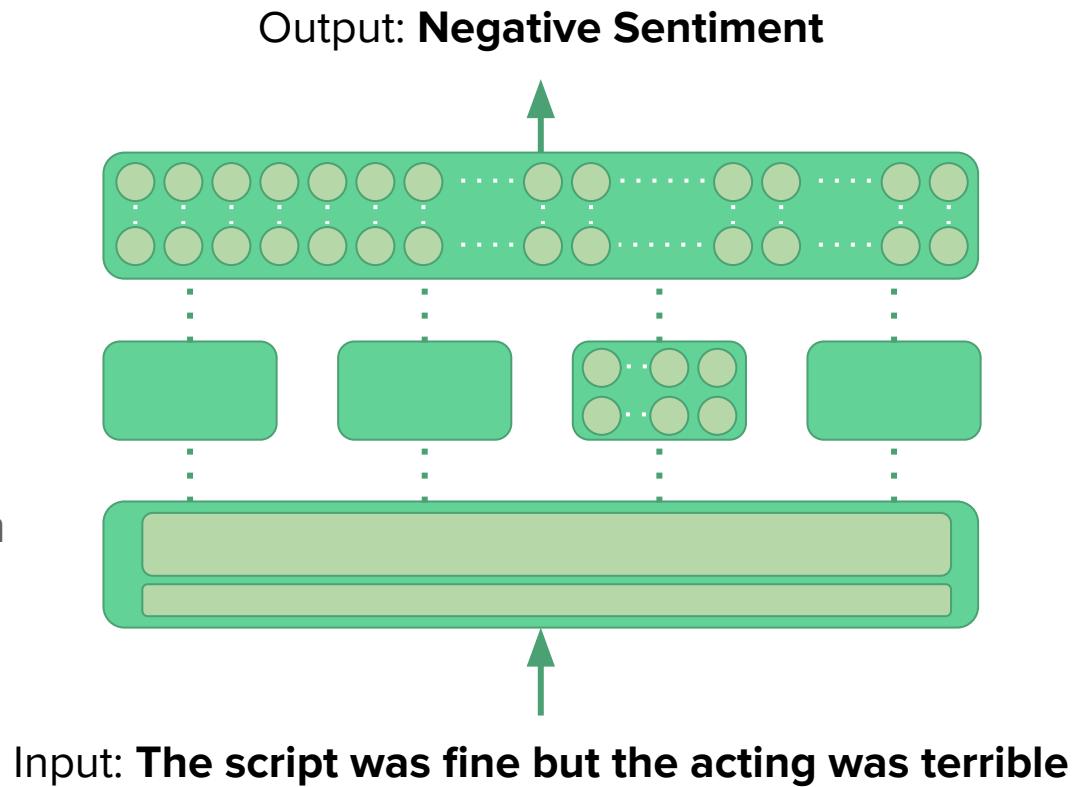
Attribution Analysis

- Methods
- Evaluation techniques
- Practical
- Case study

Concept-based Interpretation

Terminology: What is a “neuron”?

- Neural Networks may have **many components** such as attention heads, blocks, sub-layers within layers, gates/cells etc.
- Throughout this tutorial, we consider a “neuron” to be **any output dimension** from any of these components



Terminology: What is a “neuron”?

Examples of number of neurons in various components of BERT:

- An output block of BERT has **768** neurons
- The feed forward network within a BERT block has **3072** neurons
- An attention head in BERT has **64** neurons

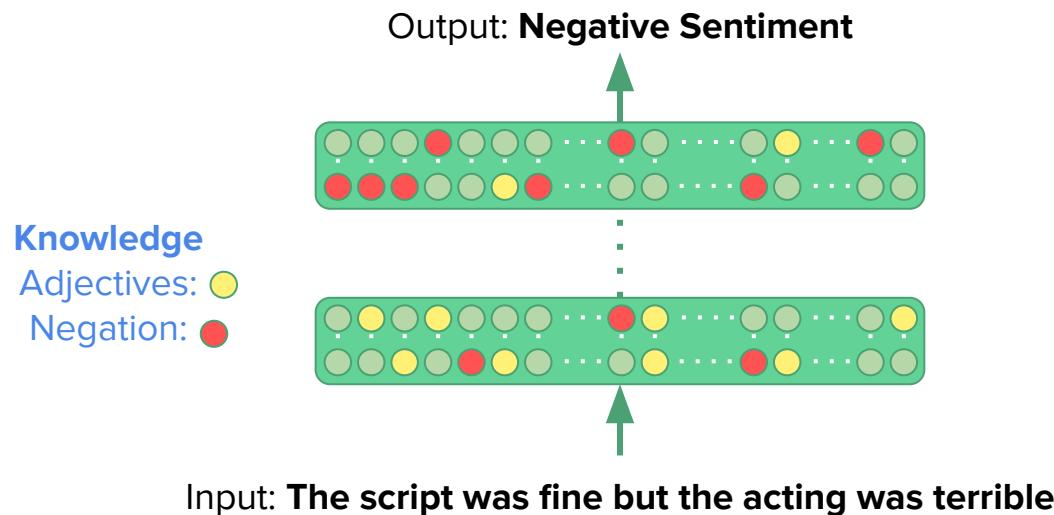
Other terminology such as “experts”, “units” and “features” have also been used interchangeably for neurons in various works

Terminology: What is a “concept”?

- **Concept** - an intrinsic property that we would like to probe in a model
- Concepts can be of various granularities
- Examples of concepts:
 - Gender, Ethnicity, Religion
 - Parts-of-speech (noun, verbs etc)
 - Semantic Concepts (location, events etc)
 - Lexical Concepts (words, morphemes, phrases)
 - Non-linguistic concepts (brackets, emoticons)
 -

Terminology: What is a “concept”?

- For example, “adjectives” and “negation” are important concepts to solve the sentiment classification task
 - Did the model learn these concepts?

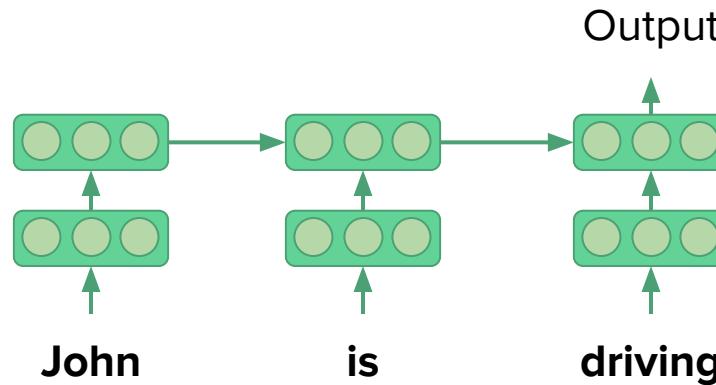


Concept Analysis

Concept Analysis

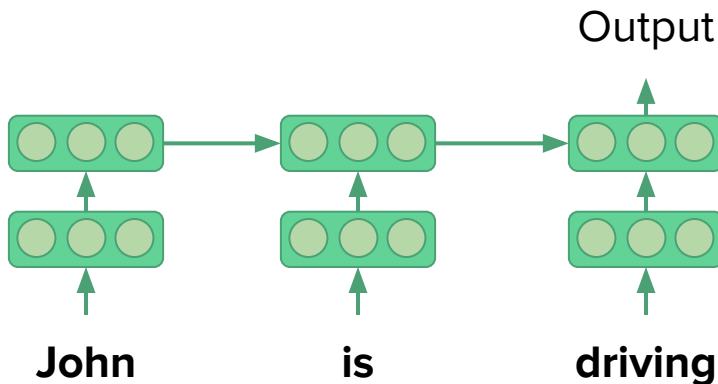
Concept analysis methods fundamentally operate on activations from a given model over a dataset

- Consider a simple model with an embedding layer and an RNN layer, each having 3 dimensions



Concept Analysis

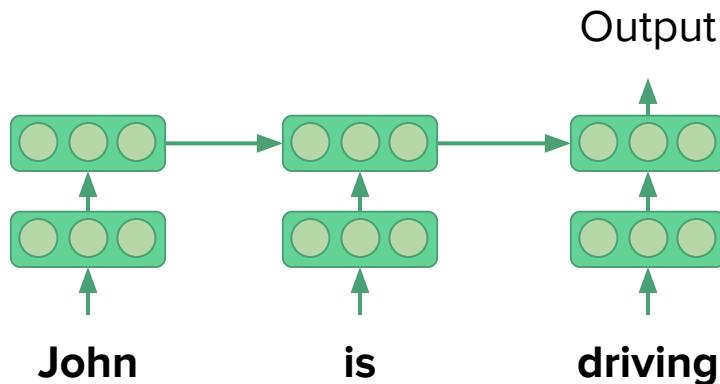
- Consider a simple model with an embedding layer and an RNN layer, each having 3 dimensions
 - Word level activations** will contain six numbers for each word in this case, **3 from the first layer** and **3 from the second layer**



Word	Activations
John	[0.1, 0.2, -0.3, 0.5, -1.2, 0.9]
Mary	[0.4, -0.5, 0.8, -0.2, 0.4, 0.4]
is	[-0.9, 1.3, -0.1, 0.1, 0.2, 0.6]
are	[-0.3, 0.5, 0.4, 0.9, -1.3, 0.3]
driving	[0.1, 0.8, -0.7, 0.0, 0.9, -0.1]

Concept Analysis

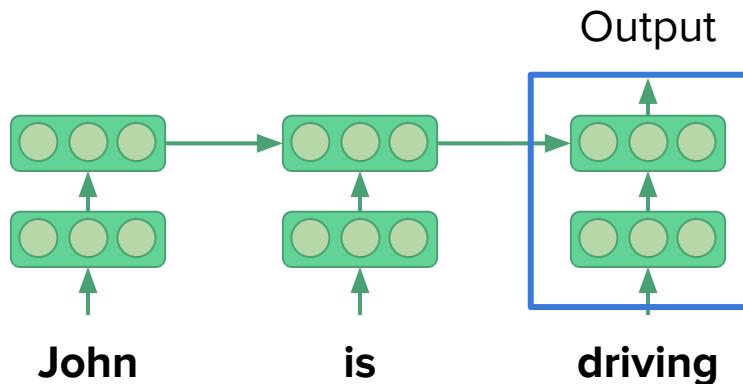
- Consider a simple model with an embedding layer and an RNN layer, each having 3 dimensions
 - Sentence level activations** will contain six numbers for each sentence



Sentence	Activations
John is driving	[0.3, -0.1, -0.3, 0.1, -0.2, 0.3]
Mary is walking	[0.8, 0.0, -0.8, -0.4, 1.2, -0.7]

Concept Analysis

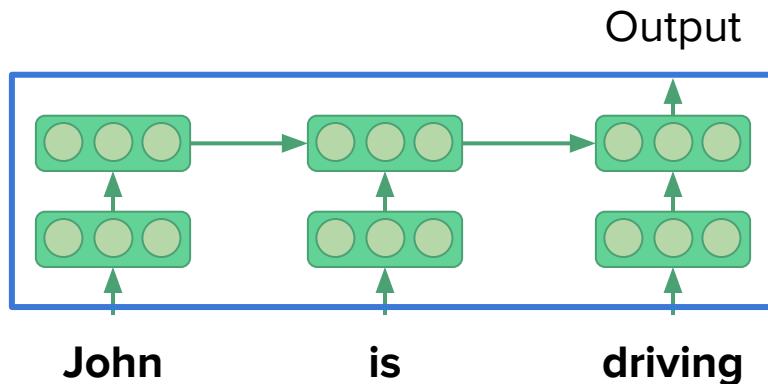
- Consider a simple model with an embedding layer and an RNN layer, each having 3 dimensions
 - Sentence level activations** will contain six numbers for each sentence
 - Activations can be obtained by considering activations of a **summary vector** (e.g. [CLS] token in transformer models, last token in left-to-right RNNs)



Sentence	Activations
John is driving	[0.3, -0.1, -0.3, 0.1, -0.2, 0.3]
Mary is walking	[0.8, 0.0, -0.8, -0.4, 1.2, -0.7]

Concept Analysis

- Consider a simple model with an embedding layer and an RNN layer, each having 3 dimensions
 - Sentence level activations** will contain six numbers for each sentence
 - Activations can be also obtained by **aggregating over word activations**, using either **average** or **max pooling** for each neuron



Sentence	Activations
John is driving	[0.3, -0.1, -0.3, 0.1, -0.2, 0.3]
Mary is walking	[0.8, 0.0, -0.8, -0.4, 1.2, -0.7]

Road Map

Concept Analysis

- **Methods**
 - Visualization
 - Corpus selection methods
 - Neuron Probing methods
 - Unsupervised methods
- **Evaluation techniques**
 - Qualitative analysis
 - Visualization
 - Corpus selection/generation
 - Quantitative analysis
 - Ablation
 - Classifier performance
 - Information theoretic methods

- **Findings**
- **Datasets**
- **Practical**

Concept Analysis

Methods

Methods

Visualization
Corpus-Selection
Neuron Probing
Unsupervised

Methods

Visualization

Corpus-Selection

Neuron Probing

Unsupervised

Visualization Methods

Visualizing neuron activations with respect to inputs is one way to identify their role

The Suez canal featured a single-lane waterway with passing locations in the Ballah Bypass and the Great Bitter Lake. It contained , according to Alois Negrelli's plans , no lock systems , with seawater flowing freely through it . In general , the water in the canal north of the Bitter Lakes flows north in winter and south in summer . South of the lakes , the current changes with the tide at Suez .

Visualization Methods

Visualizing neuron activations with respect to inputs is one way to identify their role

The Suez canal featured a single-lane waterway with passing locations in the Ballah Bypass and the Great Bitter Lake . It contained , according to Alois Negrelli's plans , no lock systems , with seawater flowing freely through it . In general , the water in the canal north of the Bitter Lakes flows north in winter and south in summer . South of the lakes , the current changes with the tide at Suez .

Comma Neuron

Visualization Methods

Visualizing neuron activations with respect to inputs is one way to identify their role

The Suez canal featured a single-lane waterway with passing locations in the Ballah Bypass and the Great Bitter Lake . It contained , according to Alois Negrelli's plans , no lock systems , with seawater flowing freely through it . In general , the water in the canal north of the Bitter Lakes flows north in winter and south in summer . South of the lakes , the current changes with the tide at Suez .

Article Neuron

Visualization Methods

Visualizing neuron activations with respect to inputs is one way to identify their role

The Suez canal featured a single-lane waterway with passing locations in the Ballah Bypass and the Great Bitter Lake . It contained , according to Alois Negrelli's plans , no lock systems , with seawater flowing freely through it . In general , the water in the canal north of the Bitter Lakes flows north in winter and south in summer . South of the lakes , the current changes with the tide at Suez .

Position Neuron

Visualization Methods

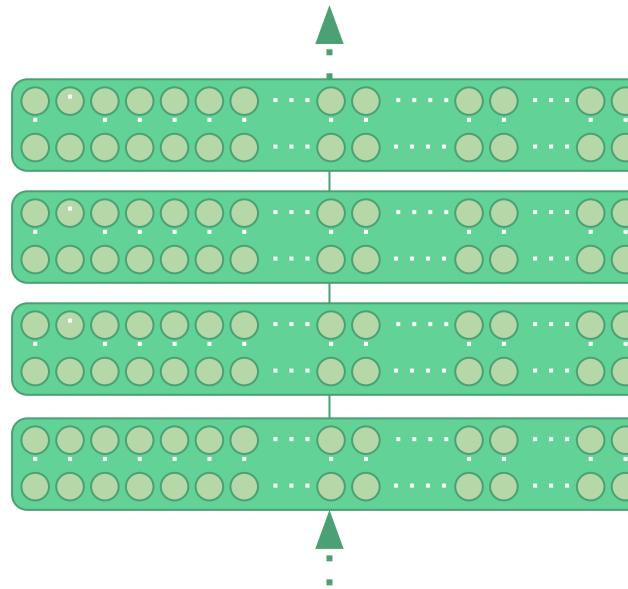
- Network consists of a large number of neurons e.g. BERT-base has 768 neurons in the output of a one transformer block
- Visualizing all these neurons is cumbersome
- A few tricks has been suggested in literature
 - Visualize saturated neurons
 - Look at neurons with either high variance or low variance
 - In case of ReLU, ignore the dead neurons

Visualization Methods

Limitations:

- Cumbersome to visualize a large number of neurons

Many layers in deep models, and
many neurons within each layer



Visualization Methods

Limitations:

Is this a verb neuron?
or a year neuron?

- Difficult to analyze polysemous neurons

Jobs regained leadership status, becoming CEO in September 1997. Apple swiftly returned to profitability under the revitalizing "Think different" campaign, rebuilding Apple's status by launching the iMac and iPod, opening a retail chain of Apple Stores in 2001, and acquiring numerous companies to broaden the software portfolio. The company was renamed to Apple Inc. in 2007, reflecting a focus toward consumer electronics, and launched the iPhone to critical acclaim and financial success. In August 2011, Jobs resigned as CEO due to health complications, and Tim Cook became the new CEO. Two months later, Jobs died, marking the end of an era for the company. In June 2019, Jony Ive, Apple's CDO, left the company to start his own firm but stated he would work with Apple as its primary client.

Visualization Methods

Limitations:

- Prone to confirmation bias

Jobs regained leadership status, becoming CEO in September 1997. Apple swiftly returned to profitability under the revitalizing "Think different" campaign, rebuilding Apple's status by launching the iMac and iPod, opening a retail chain of Apple Stores in 2001, and acquiring numerous companies to broaden the software portfolio. The company was renamed to Apple Inc. in 2007, reflecting a focus toward consumer electronics, and launched the iPhone to critical acclaim and financial success. In August 2011, Jobs resigned as CEO due to health complications, and Tim Cook became the new CEO. Two months later, Jobs died, marking the end of an era for the company. In June 2019, Jony Ive, Apple's CDO, left the company to start his own firm but stated he would work with Apple as its primary client.

Visualization Methods

Limitations:

- Prone to confirmation bias

Jobs regained leadership status, becoming CEO in September 1997. Apple swiftly returned to profitability under the revitalizing "Think different" campaign, rebuilding Apple's status by launching the iMac and iPod, opening a retail chain of Apple Stores in 2001, and acquiring numerous companies to broaden the software portfolio. The company was renamed to Apple Inc. in 2007, reflecting a focus toward consumer electronics, and launched the iPhone to critical acclaim and financial success. In August 2011, Jobs resigned as CEO due to health complications, and Tim Cook became the new CEO. Two months later, Jobs died, marking the end of an era for the company. In June 2019, Jony Ive, Apple's CDO, left the company to start his own firm but stated he would work with Apple as its primary client.

Visualization Methods

Limitations:

- Prone to confirmation bias

Jobs regained leadership status, becoming CEO in September 1997. Apple swiftly returned to profitability under the revitalizing "Think different" campaign, rebuilding Apple's status by launching the iMac and iPod, opening a retail chain of Apple Stores in 2001, and acquiring numerous companies to broaden the software portfolio. The company was renamed to Apple Inc. in 2007, reflecting a focus toward consumer electronics, and launched the iPhone to critical acclaim and financial success. In August 2011, Jobs resigned as CEO due to health complications, and Tim Cook became the new CEO. Two months later, Jobs died, marking the end of an era for the company. In June 2019, Jony Ive, Apple's CDO, left the company to start his own firm but stated he would work with Apple as its primary client.

Visualization Methods

Limitations:

- Not all concepts are human interpretable

Jobs regained leadership status, becoming CEO in September 1997. Apple swiftly returned to profitability under the revitalizing "Think different" campaign, rebuilding Apple's status by launching the iMac and iPod, opening a retail chain of Apple Stores in 2001, and acquiring numerous companies to broaden the software portfolio. The company was renamed to Apple Inc. in 2007, reflecting a focus toward consumer electronics, and launched the iPhone to critical acclaim and financial success. In August 2011, Jobs resigned as CEO due to health complications, and Tim Cook became the new CEO. Two months later, Jobs died, marking the end of an era for the company. In June 2019, Jony Ive, Apple's CDO, left the company to start his own firm but stated he would work with Apple as its primary client.

Methods

Visualization
Corpus-Selection
Neuron Probing
Unsupervised

Corpus Selection Methods

Corpus selection methods either identify **sentences that maximize a given neuron's activation**, or identify **salient neurons based on some statistics** computed over all sentences in the input corpus

- Corpus Rank
 - Rank sentences based on their informativeness for a given neuron
- Corpus Generation
 - Generate optimal sentences for a neuron
- Mask-based Neuron Selection
 - Compute statistics from the corpus with respect to concepts and neurons

Corpus Rank using Synthetic Sentences

- **Method:**
 - Instead of manual identification of concepts by visualization, split the corpus into lexical concepts of varying granularity such as, words, phrases, sentences
 - “John is sitting in the car” can be broken down into several “concepts” (e.g. based on a parse tree):
{"John", "is", "sitting", "in", "the", "car", "the car", "sitting in the car", "John is sitting in the car"}

Corpus Rank using Synthetic Sentences

- **Method:**
 - For each concept, create a synthetic sentence of average sentence length representing the particular concept only:

Original sentence: John is sitting in the car.

Concept 1: the car

Synthetic sentence: the car the car the car the car the car

Concept 2: sitting in the car

Synthetic sentence: sitting in the car sitting in the car

- Neuron that activates largely on the synthetic sentence is responsible for the concept
- Neurons are ranked by their activation values

Corpus Rank

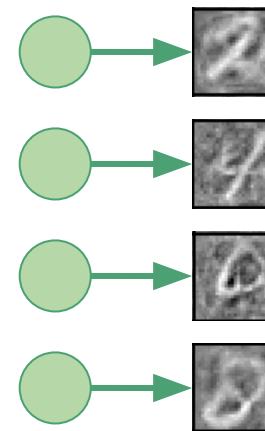
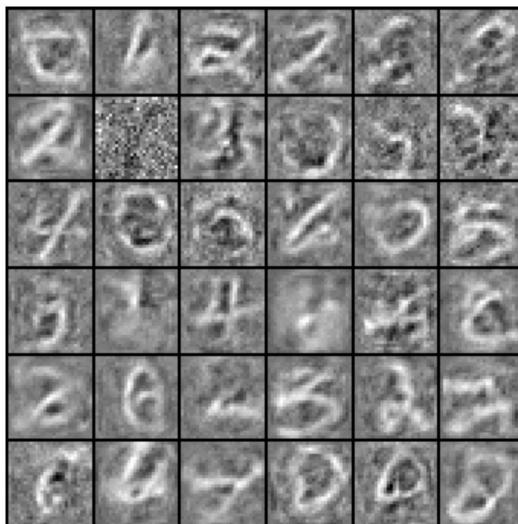
- **Limitations:**

- Corpus rank is limited to the space of a given corpus
- What if a neuron represents a diverse concept not present in the corpus?
- Neuron behaviour on unseen or ungrammatical sentences will be unexplored

Corpus Generation using Gradient Ascent

- **Method:**

- Gradient ascent is one method to **generate an input** that maximize the activations of a neuron



Corpus Generation using Gradient Ascent

- **Method:**
 - Gradient ascent cannot be directly applied to NLP because of non-continuous inputs
 - Gradient ascent with Gumble softmax solves the issue
 - It outputs a “spiky” 5-hot vector corresponding to a 5-gram which maximally activate the given neuron

method	optimal 5-gram	target neuron activation
crp	finish line at a horse	9.45
gbl	horsed horseback motocycles enthusiast they	13.32
522nd neuron (“race”) in language model projection layer		

Corpus Generation using Gradient Ascent

- **Limitations:**

- Limited exploration has been done in NLP using gradient ascent with gumble
- The generated words/phrases may not make sense

Mask-based Neuron Selection

- **Method:**
 - Compare binary activation masks with concept masks to determine important neurons with respect to the concept
 - **Neuron mask creation** - a binary mask of each neuron based on some threshold on the activation values across the corpus
 - **Concept mask creation** - a binary mask of each concept based on its presence or absence across the corpus
 - **Intersection-over-union (IoU)** of all neuron masks and a concept mask
 - Rank neurons by their IoU with respect to the concept

Mask-based Neuron Selection



Mask-based Neuron Selection

Sentences

Corpus	Noun
I am going to Paris by train	1
Are you there?	0
Beautiful cars	1
Ugly cars	1

Concept mask can be created based on the
presence of a given part-of-speech tag

Mask-based Neuron Selection

Sentences

Corpus	Noun	Verb
I am going to Paris by train	1	1
Are you there?	0	1
Beautiful cars	1	0
Ugly cars	1	0

Concept mask can be created based on the
presence of a given part-of-speech tag

Mask-based Neuron Selection

Sentences

Corpus	Noun	Verb	Paris
I am going to Paris by train	1	1	1
Are you there?	0	1	0
Beautiful cars	1	0	0
Ugly cars	1	0	0

Concept mask can be created based on the
presence of particular tokens

Mask-based Neuron Selection

Sentences

Corpus	Noun	Verb	Paris	Neuron 1	Neuron 2
I am going to Paris by train	1	1	1	0.7 (1)	-0.5 (0)
Are you there?	0	1	0	-1.2 (0)	0.9 (1)
Beautiful cars	1	0	0	0.7 (1)	-1.2 (0)
Ugly cars	1	0	0	0.0 (0)	0.4 (1)

Neuron masks can be created based on a
threshold on the activation values (e.g. 0)

Mask-based Neuron Selection

Sentences ↓

Corpus	Noun	Verb	Paris	Neuron 1	Neuron 2
I am going to Paris by train	1	1	1	0.7 (1)	-0.5 (0)
Are you there?	0	1	0	-1.2 (0)	0.9 (1)
Beautiful cars	1	0	0	0.7 (1)	-1.2 (0)
Ugly cars	1	0	0	0.0 (0)	0.4 (1)

IoU: $\frac{2}{3}$
Noun

IoU: $\frac{1}{3}$
Verb

IoU: $\frac{1}{2}$
Paris

IoU: $\frac{1}{4}$
Noun

IoU: $\frac{1}{3}$
Verb

IoU: $0/3$
Paris

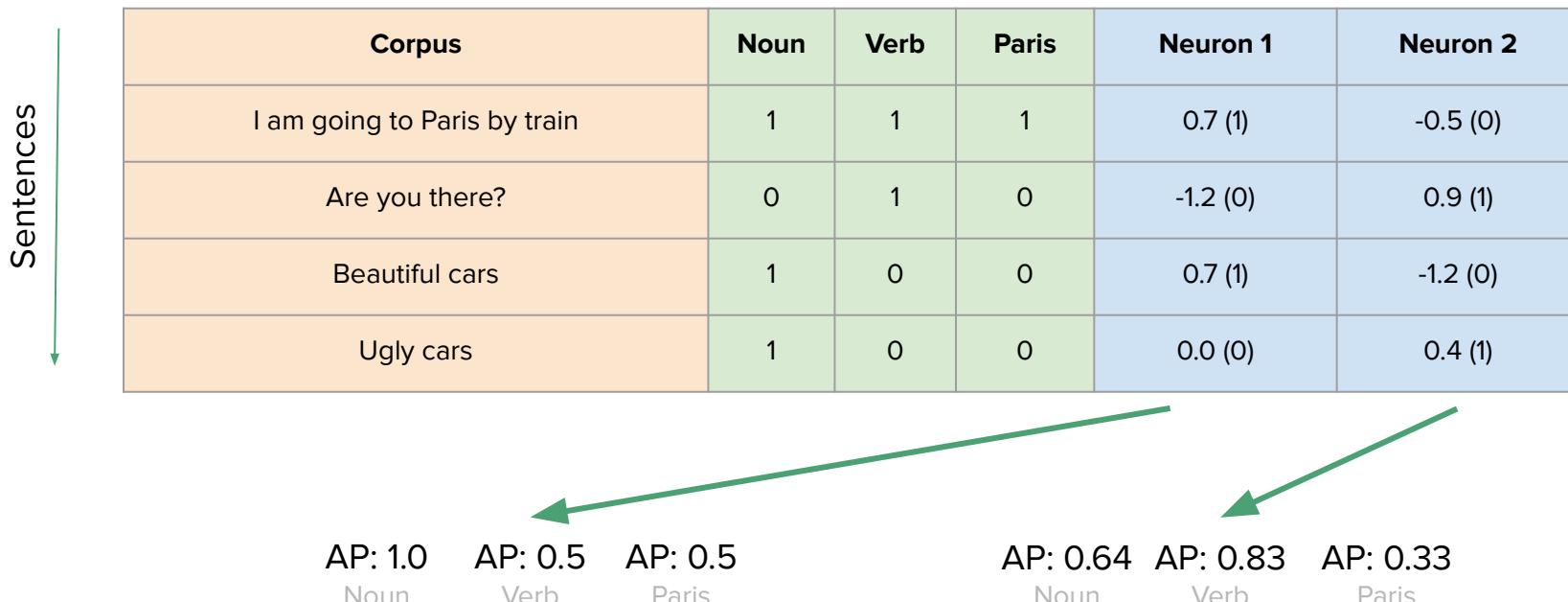
Mask-based Neuron Selection

Instead of considering binary masks from neurons, we can use the *activation values* themselves as prediction scores

- **Method:**
 - Compute average precision score per neuron and per concept
 - Rank neurons by their average precision score with respect to the concept

Mask-based Neuron Selection

Ranking by average precision score per neuron and per concept

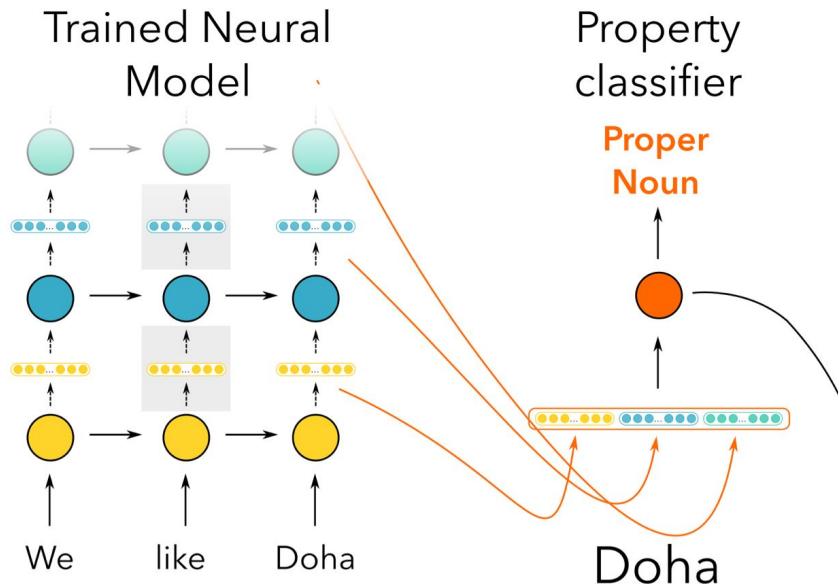


Methods

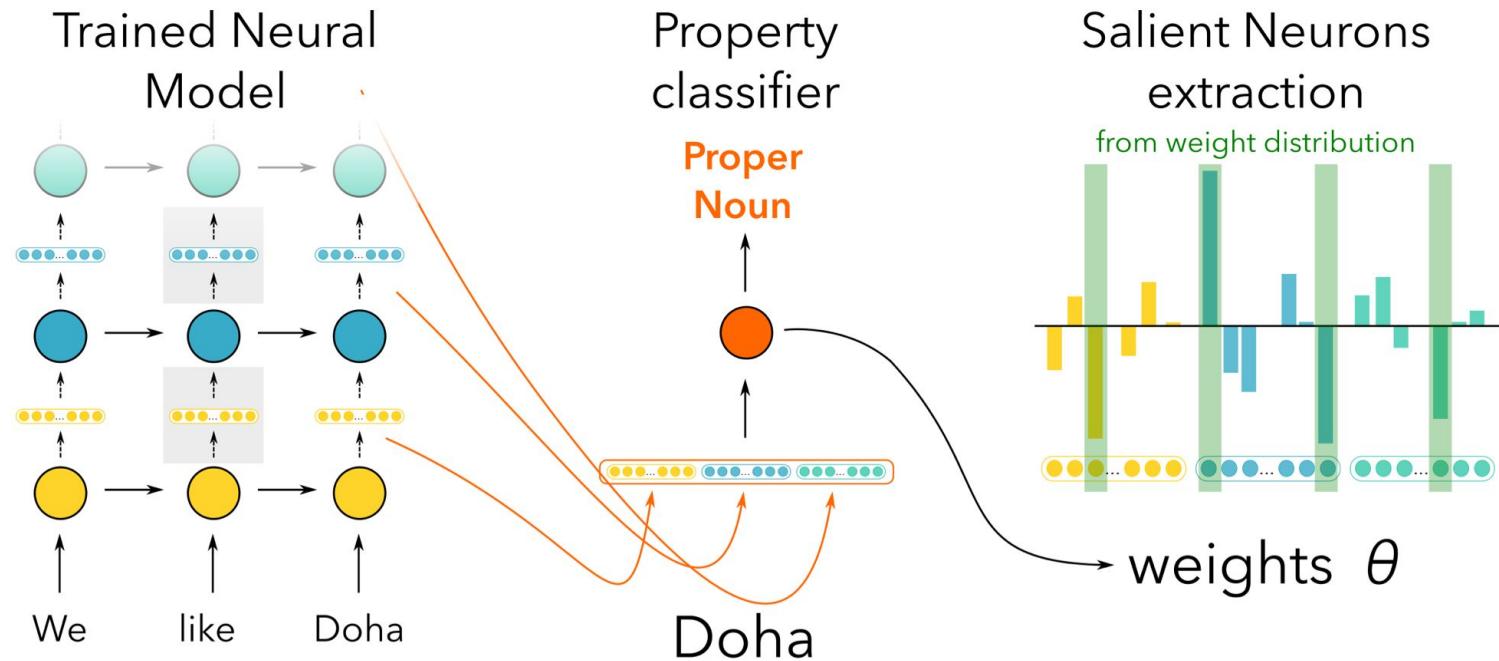
Visualization
Corpus-Selection
Neuron Probing
Unsupervised

Neuron Probing Methods

- **Method:**
 - Given supervised data for a concept (e.g. word → is_Noun), extract activations of all neurons with respect to input words
 - Train a classifier for the concept of interest using neuron activations as features
 - Derive a ranking of neurons from the trained classifier



Neuron Probing by Linear Classifier



Neuron Probing by Linear Classifier

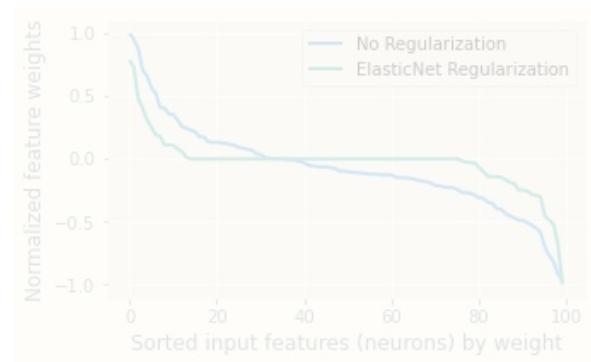
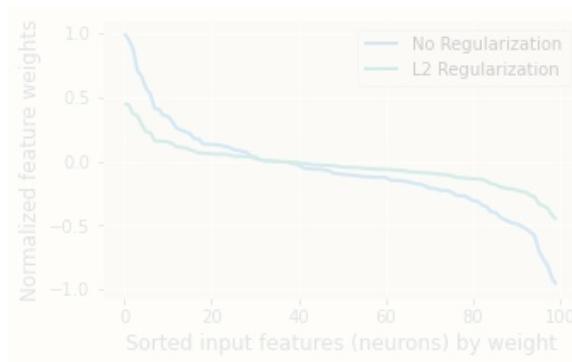
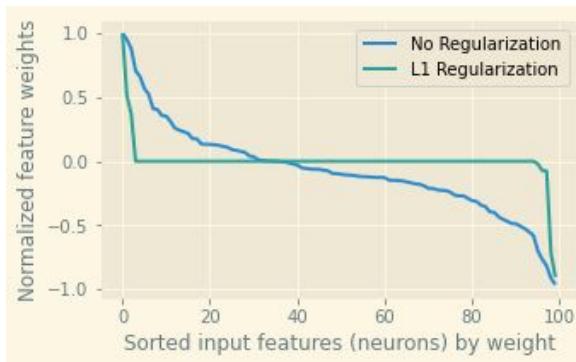
- **Method:**

- Feature weights are used as a proxy for importance of a input feature (neuron in our case)
- The choice of regularization directly affects the neuron ranking
- L1 (Lasso), L2 (Ridge) and ElasticNet (L1+L2) regularization has been explored

Neuron Probing by Linear Classifier

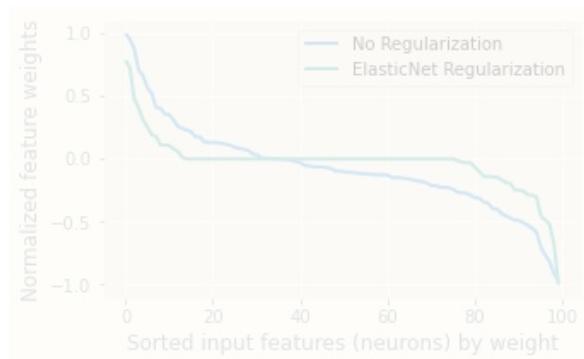
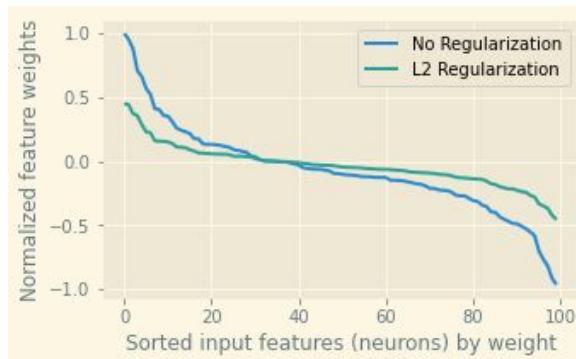
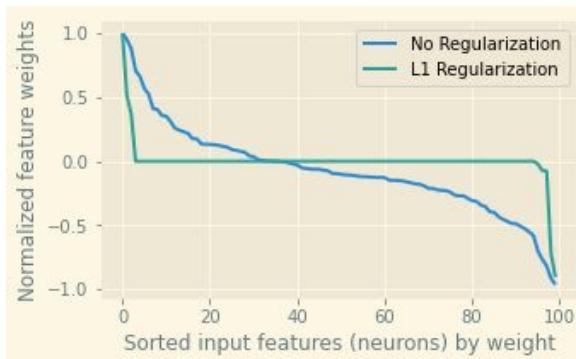
- **Method:**

- L1 (Lasso) regularization results in few top neurons
- Induces sparsity by zeroing out less important features



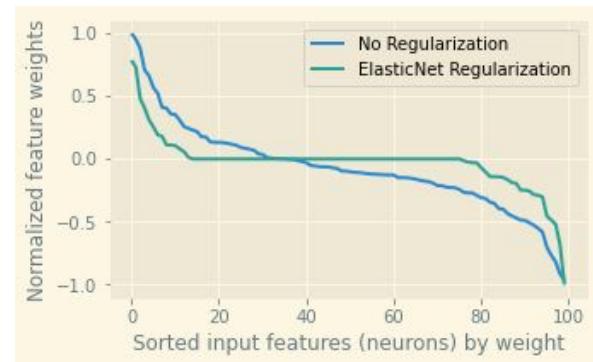
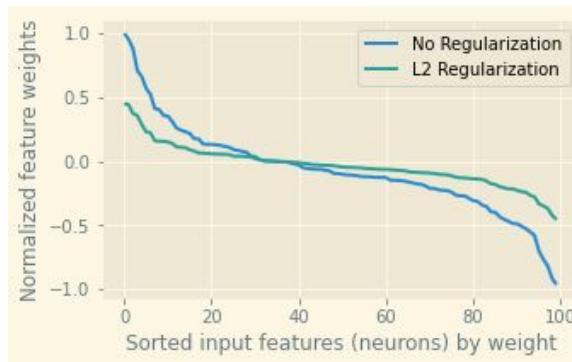
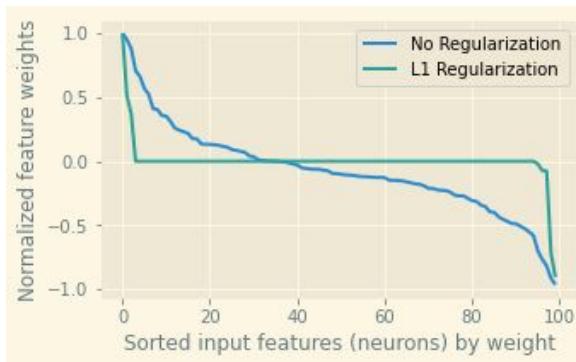
Neuron Probing by Linear Classifier

- **Method:**
 - L2 (Ridge) regularization encourages neuron grouping



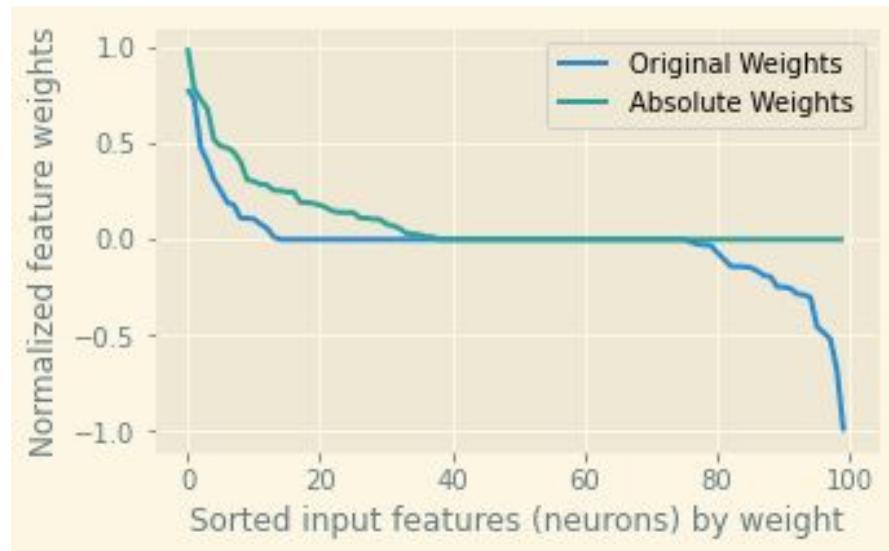
Neuron Probing by Linear Classifier

- **Method:**
 - ElasticNet (L1+L2) regularization aims to combine the best of both worlds



Neuron Probing by Linear Classifier

- The absolute feature weight is also commonly considered while deriving the ranking
- This takes into account neurons with both positive and negative weights



Neuron Probing by Random Forest Classifier

- **Method:**

- Instead of using a Linear classifier, a Random Forest classifier can also be used to rank neurons
- **Gini Importance** is used to rank each input feature (neuron) by its purity, which essentially measures how often an input feature was used to split a node in one of the trees

Neuron Probing Methods

Important considerations:

- Majority class
 - Tag all test instances with the majority class
- Randomly initialized representations
- Control tasks
- Overall performance
- Size of the concept dataset

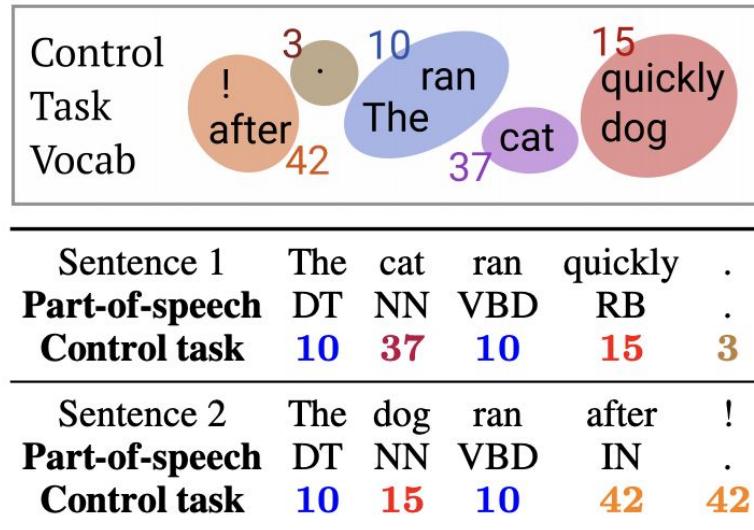
Randomly Initialized Representations

A probe may be able to sufficiently extract enough signal from a large number of neurons and achieve high performance, regardless of whether the concept is actually represented by any neuron

- Train classifier using random initialization
- The performance of this probe must be lower than the original probe

Control Tasks

- **Intuition:** Break the linguistic relationship between words in the task
- Associate each word type with random labels, i.e. controlled dataset
 - Make labels independent of the linguistic property of a word
- Controlled dataset has the same input and output space as that of the linguistic task



Control Tasks

- **Method:**
 - Train a classifier on the controlled dataset
 - **Selectivity** = difference in performance of the classifier training using original POS dataset vs. controlled dataset
 - High value of selectivity reflects a good probe i.e. representation has learned the linguistic property

Overall Performance

- The overall performance of the probe classifier should be substantially higher than majority class, random representations and control tasks to ensure a trustworthy ranking

Neuron Probing Methods

- **Limitations:**
 - Need of supervised data for every concept
 - Understudied concepts must be pre-defined
 - Choice of probe directly affects the final ranking
 - Classifier (linear, random forest)
 - Regularization (l_1 , l_2 , elastic, etc)
 - Parameters (training hyperparameters etc)

Methods

Visualization
Corpus-Selection
Neuron Probing
Unsupervised

Unsupervised Methods

- Neuron probing aims to surface knowledge within a model **with respect to a concept**
- However, it is dependent on already defined concepts with **supervised data**

Unsupervised Methods

Unsupervised methods aim to analyze underlying knowledge patterns in the models without any supervised data

- These patterns can then be further analyzed either by automatic means like weak annotations or having human in the loop
- This class of methods targets questions like:
 - What are important neurons with respect to the model?
 - Can we extract the encoded knowledge without concept annotations?

Unsupervised Methods

- Ablation
- Gaussian-based probing
- Matrix Factorization
- Clustering methods
- Multi-model search

Ablation Methods

- Ablation methods mask/erase neuron(s) of the network to understand their contribution
 - A drop in performance reflects that the neuron(s) are critical
- A common way to ablate a neuron is to clamp its value to **zero or it's mean value**

Single Neuron Ablation

- **Method:**

- Ablate exactly one neuron at a time independently
- Compare the difference in confidence of a particular class or model in general
- Rank the neurons by the difference

Single Neuron Ablation

- **Method:**

- Ablate exactly one neuron at at time independently
- Compare the difference in confidence of a particular class or model in general
- Rank the neurons by the difference
- Comparing raw confidence difference may not be ideal
 - Normalized confidence difference has been proposed as another metric to use for ranking

Single Neuron Ablation

- **Method:**

- Ablate exactly one neuron at a time independently
- Compare the difference in confidence of a particular class or model in general
- Rank the neurons by the difference
- Comparing raw confidence difference may not be ideal
 - Normalized confidence difference has been proposed as another metric to use for ranking

- **Limitations:**

- The effect of a single neuron may be negligible in large models
- Importance scores obtained from very small differences may not reflect the true ranking

Multiple Neuron Ablation - Greedy Ablation

- **Method:**
 - Ablate all neurons individually and rank them in the order of drop in performance
 - Select the neuron with the largest drop in performance
 - Reiterate the first two steps with N-1 neurons
- **Limitations:**
 - Order of ablation is critical
 - Ablating in all possible permutations is an NP-complete problem
 - Ignores the interplay between different neurons

Unsupervised Methods

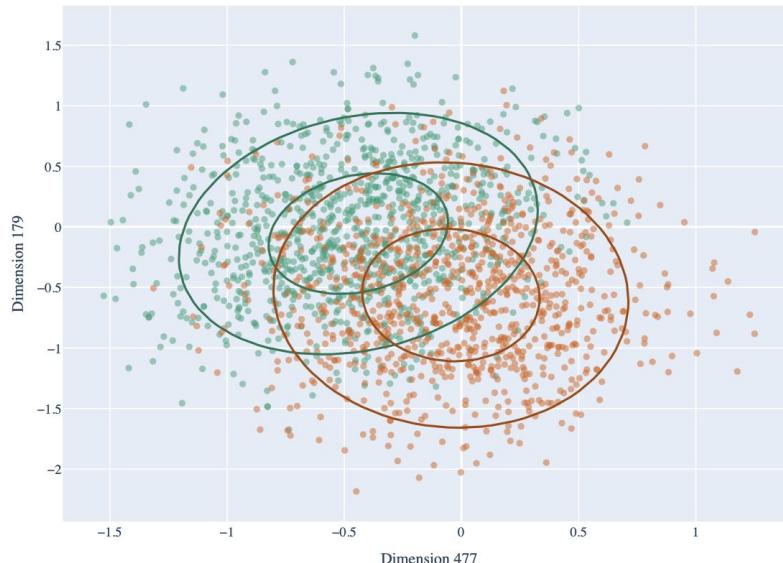
- Ablation
- Gaussian-based probing
- Matrix Factorization
- Clustering methods
- Multi-model search

Gaussian-based Probing

- **Fundamental assumption:** Neuron activations follow a gaussian distribution w.r.t. specific concepts.

Gaussian-based Probing

- **Fundamental assumption:** Neuron activations follow a gaussian distribution w.r.t. specific concepts.



Past and Present tense
activation patterns
across two neurons
follow a gaussian like
distribution

Gaussian-based Probing

- **Method:**
 - Fit a multivariate gaussian over all neurons across a dataset
 - Since a multivariate gaussian is decomposable, “Probes” for any subset of neuron(s) can be extracted from this in constant time
 - Use extracted probes and their associated distributions to conduct analysis
 - Use maximum a posteriori (MAP) estimate to greedily search for top neurons that predict some concept of interest with high accuracy and mutual information
- **Limitations:**
 - Not all neuron activations can be fit using a gaussian
 - Search is still expensive

Unsupervised Methods

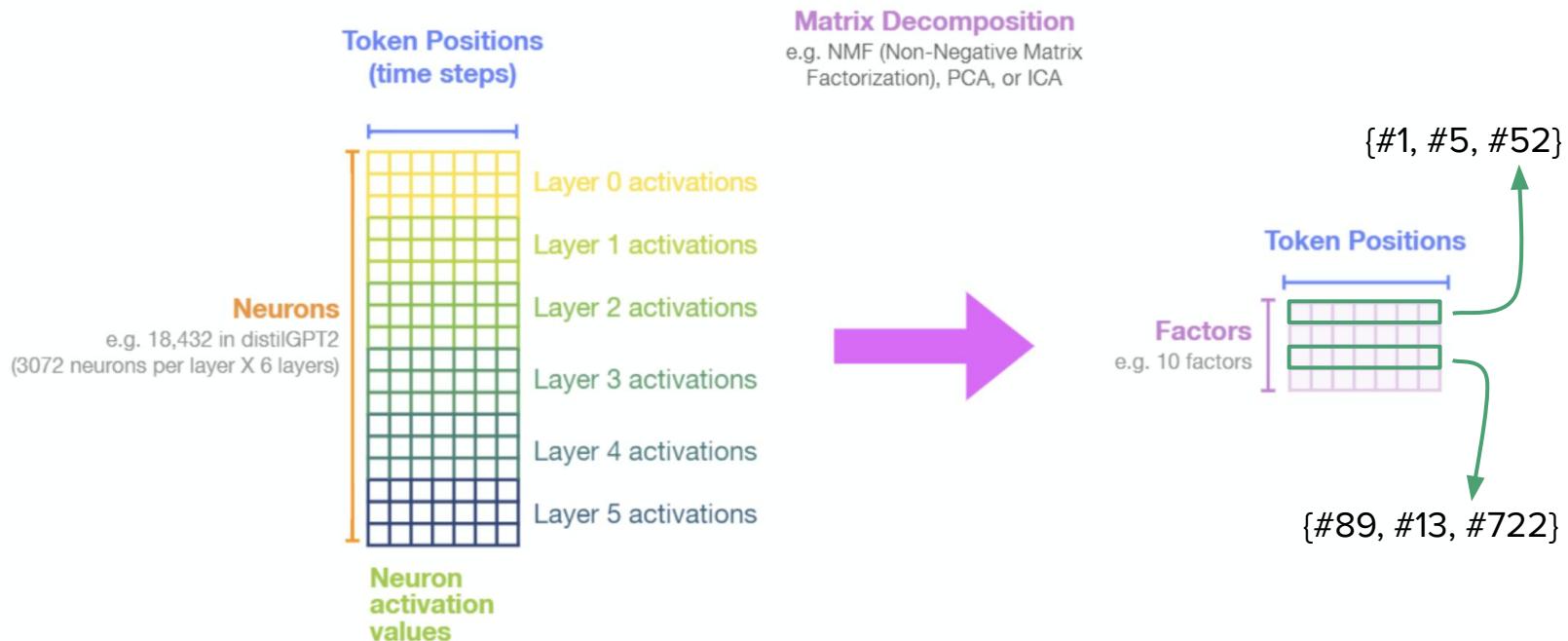
- Ablation
- Gaussian-based probing
- Matrix Factorization
- Clustering methods
- Multi-model search

Matrix Factorization

Another way to group neurons is using **matrix factorization**

- Find optimal strategies to break up large matrices (for example, neuron activations)
- Neurons performing similar functions will be grouped together

Matrix Factorization



Matrix Factorization

- **Limitations:**
 - Number of factors (“groups” of neurons) needs to be pre-defined
 - Groups of neurons are computed per-prediction, and hence some aggregation over many predictions is necessary to make broader claims
- Preliminary work: not much work done in the NLP domain

Unsupervised Methods

- Ablation
- Gaussian-based probing
- Matrix Factorization
- Clustering methods
- Multi-model search

Clustering Methods

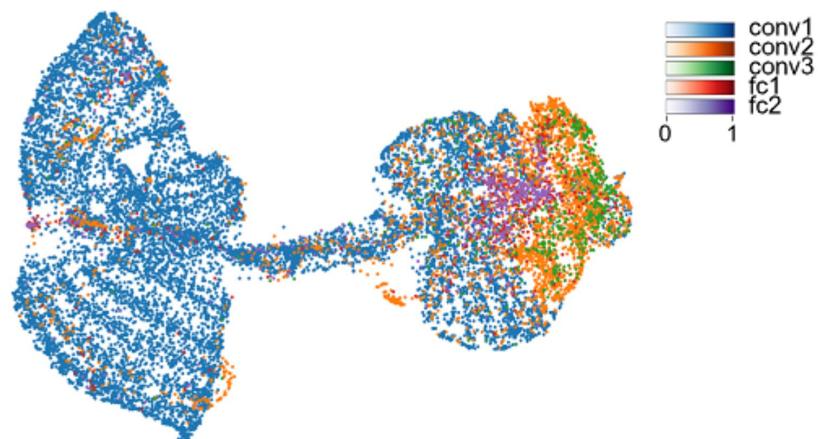
- **Fundamental assumption:** Neuron form clusters in activation space that correspond to some concepts and/or classes

Clustering Methods

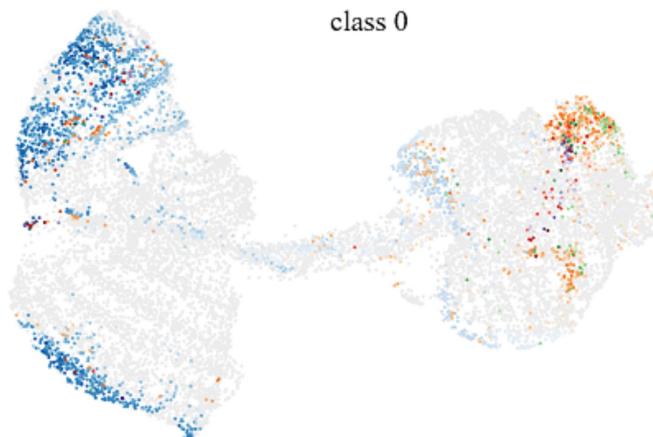
- **Method:** Activation-based clustering
 - Clustering very high dimensional representations usually does not work well
 - Use a method like UMAP to preserve clusters from high dimensional space in lower dimensional spaces
 - Analyze these lower dimensional representations manually for clusters or learn clusters automatically using some weak supervision

Clustering Methods

- **Method:** Activation-based clustering



Neurons arranged by their similarity



Neurons activating for a specific class

Clustering Methods

- **Method:** Correlation-based clustering
 - **Fundamental assumption:** Many neurons in large-pretrained models are redundant within a network
 - **Intuition:**
 - Collect activations of all neurons over a large dataset
 - Find neurons whose activation patterns over the dataset **correlate** with each other
 - Neurons with high correlation are redundant

Clustering Methods

- **Correlation-based clustering**
 - Extract neuron vectors (neuron activations over a dataset)

	Input 1	Input 2	Input 3
Neuron 1	0.9	-0.5	0.0
Neuron 2	0.4	-0.9	-0.3
Neuron 3	0.8	-0.6	0.0
Neuron 4	0.3	-0.8	-0.5

Clustering Methods

- **Correlation-based clustering**
 - Extract neuron vectors (neuron activations over a dataset)
 - Compute the Pearson product-moment correlation of every neuron vector with every other neuron

	Input 1	Input 2	Input 3
Neuron 1	0.9	-0.5	0.0
Neuron 2	0.4	-0.9	-0.3
Neuron 3	0.8	-0.6	0.0
Neuron 4	0.3	-0.8	-0.5



Neurons	n ₁	n ₂	n ₃	n ₄
n ₁	1	-0.26	0.99	-0.22
n ₂	-0.26	1.0	-0.33	0.99
n ₃	0.99	-0.33	1.0	-0.30
n ₄	-0.22	0.99	-0.30	1.0

Clustering Methods

- **Correlation-based clustering**
 - Extract neuron vectors (neuron activations over a dataset)
 - Compute the Pearson product-moment correlation of every neuron vector with every other neuron
 - Cluster the correlation matrix using **agglomerative hierarchical clustering**
 - Neurons in one cluster learn similar things

Clustering Methods

- **Limitations:**

- Number of clusters needs to be defined using hyperparameters
- Aggressive clustering may lead to dissimilar neurons being grouped together

Unsupervised Methods

- Ablation
- Gaussian-based probing
- Matrix Factorization
- Clustering methods
- Multi-model search

Multi-model Search

- **Fundamental assumption:** If multiple models (performing the same task) are learning a concept, the concept is important for the task
- Conversely, if a concept is important to solve a task, all models should learn that concept

Multi-model Search

- **Method:**
 - Find neurons across models that **behave similarly** on a given dataset
 - Similarly-behaving neurons can be thought of as learning *some concepts* important for the task
 - Similar to correlation-based grouping, but it is done across models
 - Rank the neurons based on how similar they are to neurons from other models, and analyze the top neurons in the resultant ranking

Multi-model Search

- **Method:**

- Calculate the pairwise Pearson correlation between neurons across multiple models
- Rank the neurons by aggregating over correlations across all neurons

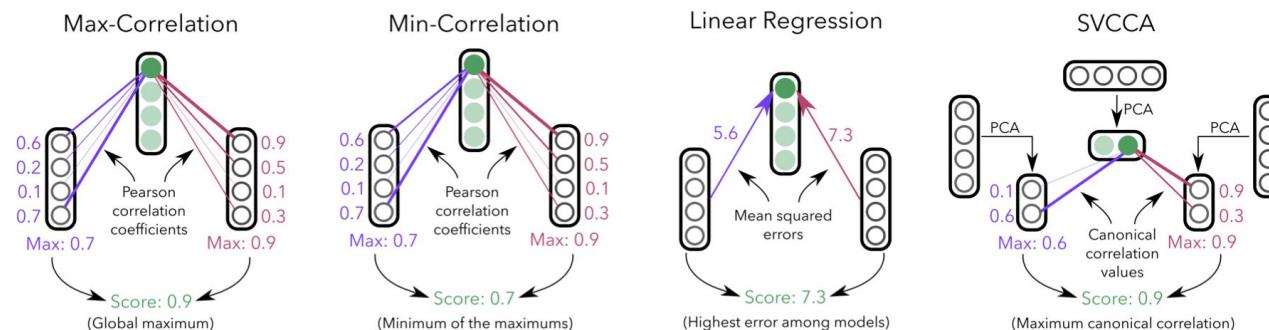
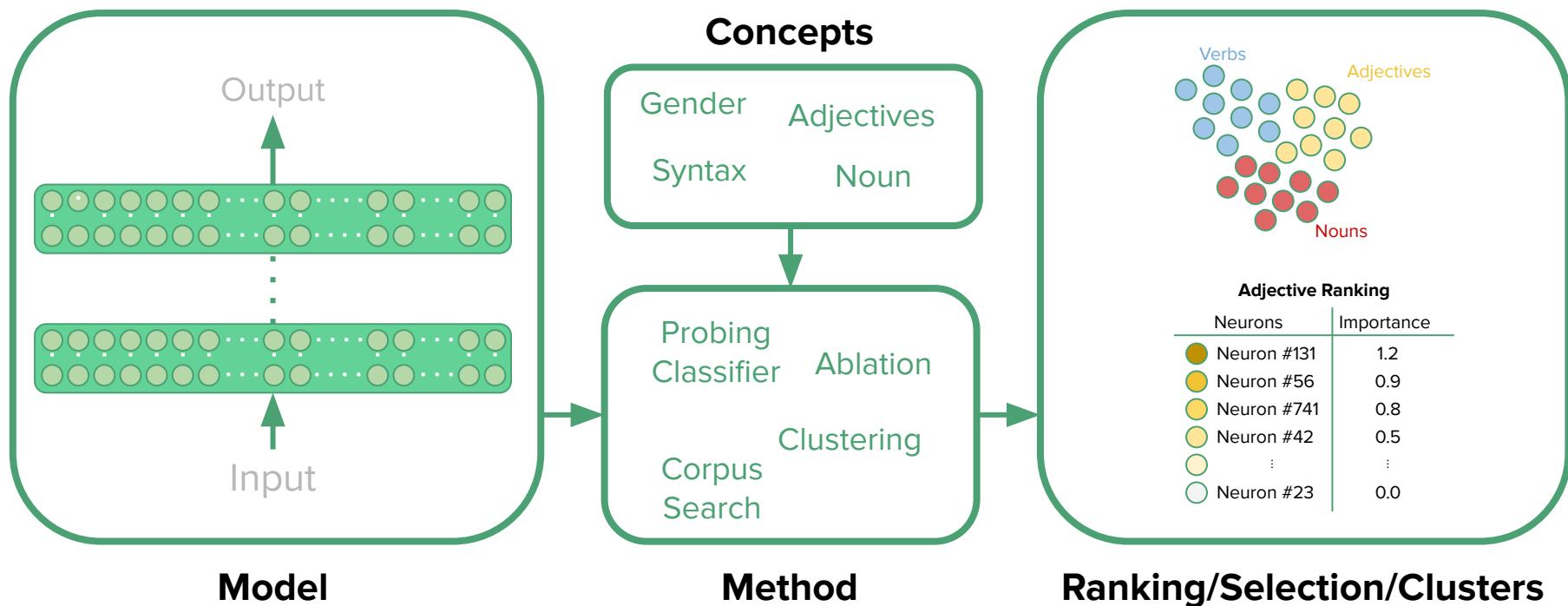


Figure 1: An illustration of the correlation methods, showing how to compute the score for one neuron using each of the methods. Here the number of models is $M = 3$, each having four neurons.

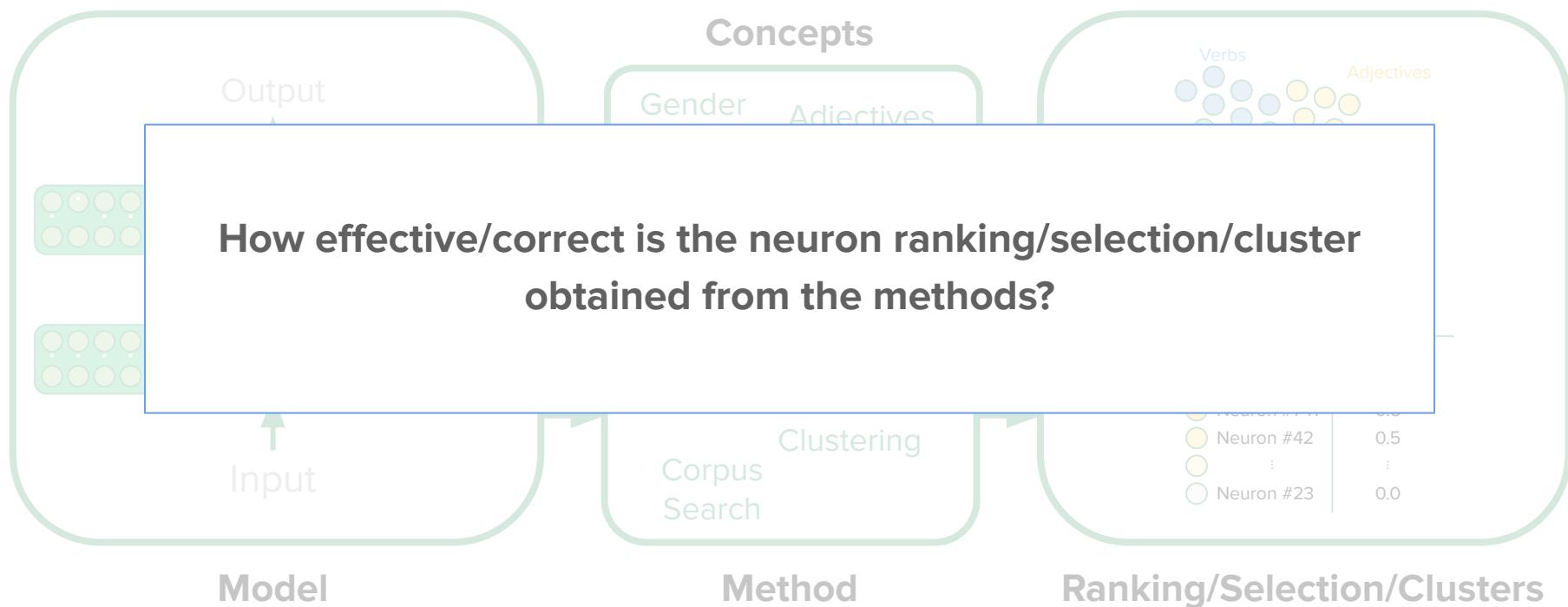
Concept Analysis

Evaluation

Evaluation Techniques



Evaluation Techniques



Evaluation

Ablation

Classification performance

Cluster comparison

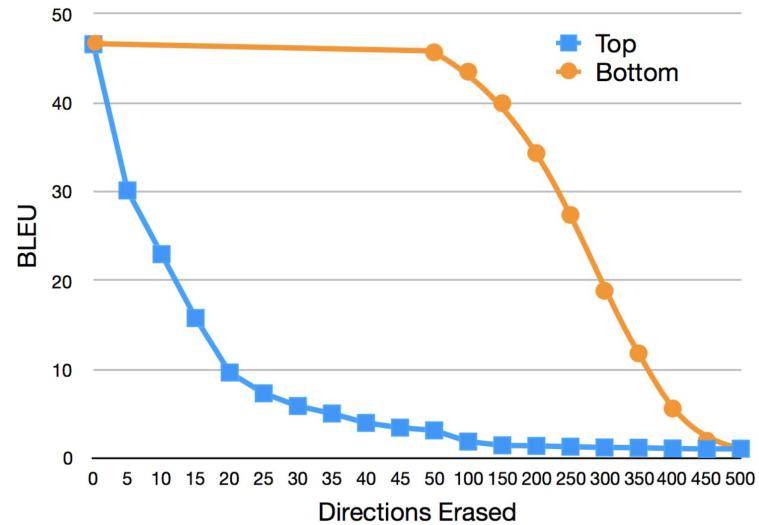
Qualitative evaluation

Ablation

Given salient list of neurons from a model, **how faithful is the neuron ranking with respect to model?**

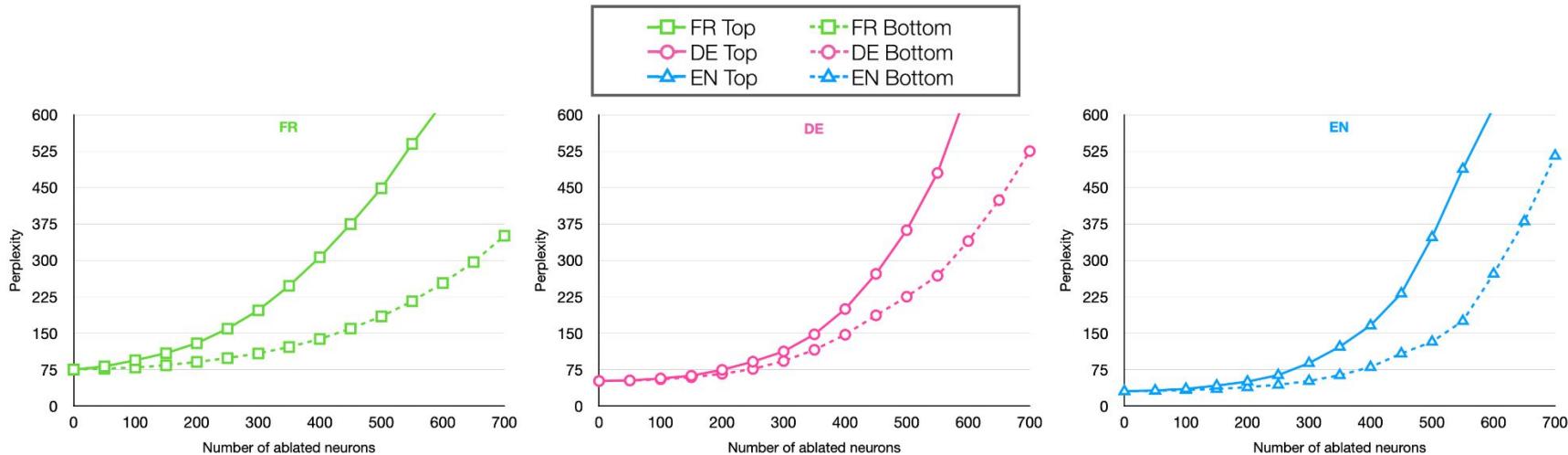
Ablation

- **Ablate top/bottom/random X% neurons** in the original model and compare the performance
- The drop in performance when ablating the top neurons should be greater compared to ablating bottom and random neurons
- The evaluation may also indirectly reflect how important a concept is, with respect to the prediction



Ablation

- Ablating neurons by their ranking order in a Language Model (**perplexity** metric for evaluation)



Evaluation

Ablation

Classification performance

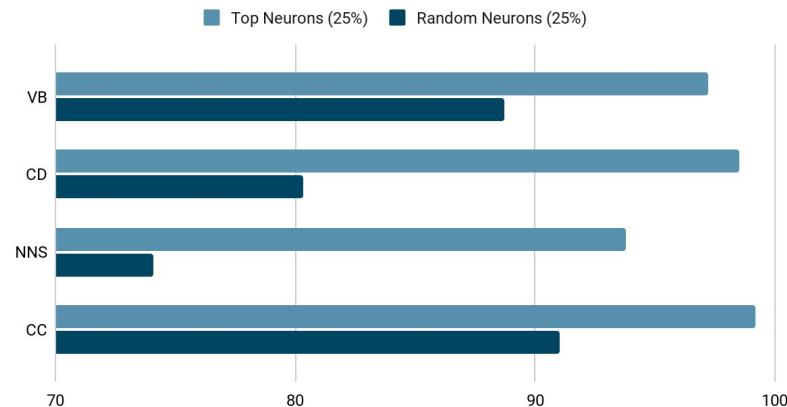
Cluster comparison

Qualitative evaluation

Classification Performance

- Given a set of neurons with respect to a concept
- Train a classifier for the concept
 - Using selected set of neurons or X% of top neurons
 - Using random set of neurons of the same size
- Difference in performance reflects whether the selected neurons represent the concept

Classifier Accuracy on Selected Neurons



Evaluation

Ablation

Classification performance

Cluster comparison

Qualitative evaluation

Cluster comparison

- Some unsupervised methods result in clusters of neurons that are learning similar information
- One way to evaluate these clusters is to see if they correspond to some ground truth clusters

Cluster comparison

- **B³**: Standard clustering evaluation metric

[Intrinsic Probing through Dimension Selection](#) (Hennigen et al. 2020)

[Asking without Telling: Exploring Latent Ontologies in Contextual Representations](#) (Michael et al. 2020)

Cluster comparison

- **B³**: Standard clustering evaluation metric
- **Normalized PMI**: Check if clustering agrees with gold labels
- **Classifier Performance**: Accuracy with classifier trained on neurons from a single cluster

[Intrinsic Probing through Dimension Selection](#) (Hennigen et al. 2020)

[Asking without Telling: Exploring Latent Ontologies in Contextual Representations](#) (Michael et al. 2020)

Cluster comparison

- **B³:** Standard clustering evaluation metric
- **Normalized PMI:** Check if clustering agrees with gold labels
- **Classifier Performance:** Accuracy with classifier trained on neurons from a single cluster
- **Qualitative Analysis:** Cluster diversity/uncertainty

[Intrinsic Probing through Dimension Selection](#) (Hennigan et al. 2020)

[Asking without Telling: Exploring Latent Ontologies in Contextual Representations](#) (Michael et al. 2020)

Evaluation

Ablation

Classification performance

Cluster comparison

Qualitative evaluation

Qualitative Evaluation

Given we have a set of selected neurons, neuron selection methods can be used to provide a qualitative evaluation

- Visualization
- Corpus selection
- Corpus generation

Limitations

- Results are not comparable across different methods
- Neurons behavior may reflect an artifact of the dataset
- Need of standard evaluation benchmarks
 - Use similar datasets to extract activations
 - Use identical concepts
 - Compare results across various datasets
 - Data dependent vs. core model specific neurons

Limitations

- Due to the distributed knowledge and complexity of the models, different neurons may be learning similar patterns and various methods select different subset of such neurons
 - Because of redundancy, two methods may bring very different rankings (but both are correct)
- Many concepts are uninterpretable
- Having a human in the loop would be helpful
- It is possible that neuron may be coming at the bottom of ranking, but may still contain some information

Concept Analysis

Findings

Applications of Neuron Probing

- Now that we have neurons, what can they be used for?
 - Analysis
 - Concept Analysis
 - Architectural Analysis
 - Redundancy Analysis
 - Applications
 - Efficient Feature Selection
 - Manipulation
 - Generating Explanations
 - Domain Adaptation
 - Network Pruning and Distillation
 - Architectural Search

Findings and Applications

Concept Discovery

Architectural Analysis

Applications

Position Neuron

LSTM Language
Model

- Neurons encode long range information
 - **Position** neuron activates
 - positively at the beginning of the sentence
 - negatively towards the end

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

Quotes Neuron

LSTM Language Model

- LSTMs neurons encode long range information
 - **Quotes** neuron
 - Activates inside quotes

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Subject-verb dependency neuron

LSTM Language Model

- One neuron that is closely involved in tracking subject-verb agreement

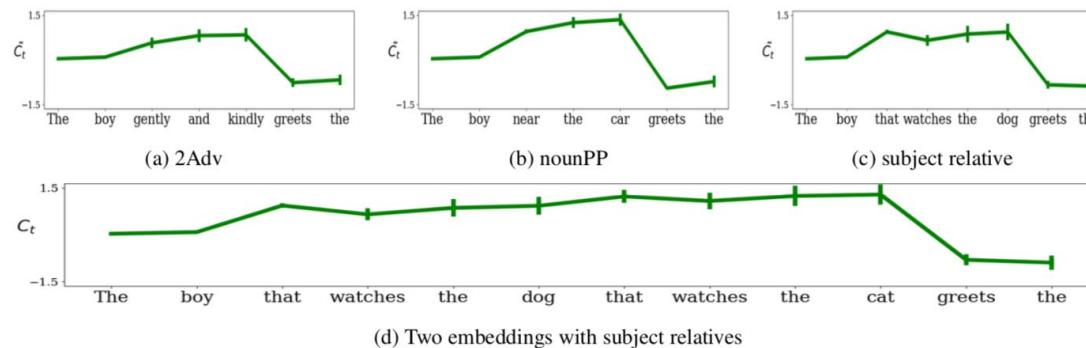
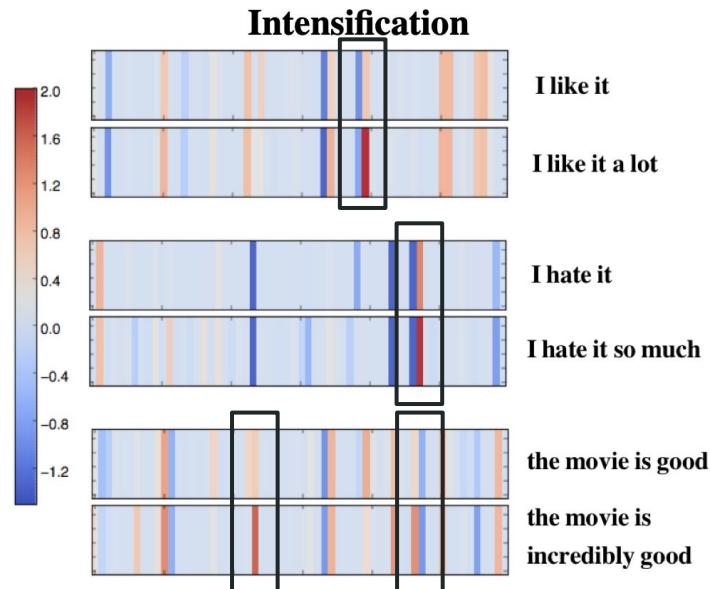


Figure 3: Cell activity of syntax unit 1150 while processing various syntactic structures. Values averaged across all stimuli in an NA-task, with error bars representing standard deviations. Relative clause NA-task stimuli were specifically generated for this visualization.

Intensifying Neuron

LSTM Language Model

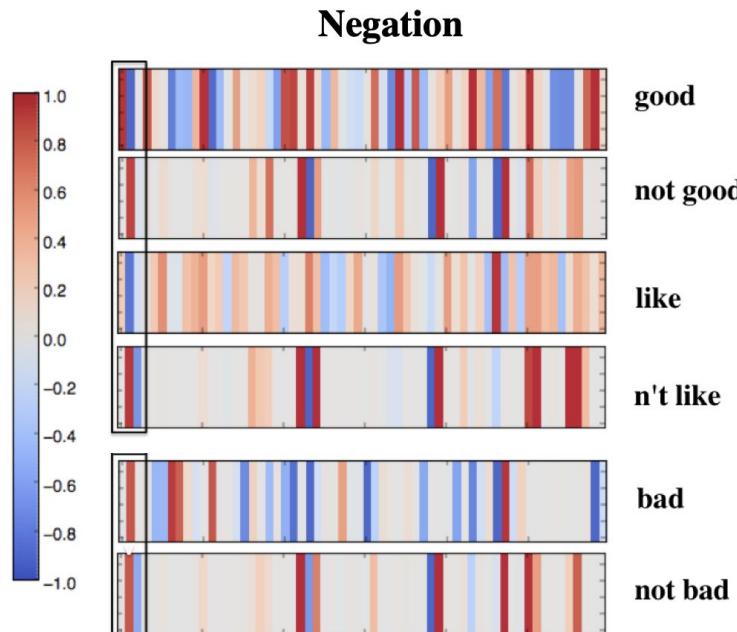
- Some neurons are “intensified” in the presence of certain tokens (“a lot”, “so much” etc)



Enhancing Negative Sentiment

LSTM Language
Model

- Some neurons are “flipped” in the presence of certain tokens (“not”)



Capturing Specific Linguistic Phenomena

LSTM Machine
Translation

Supports the efforts of the Libyan authorities to recover
funds misappropriated under the Qadhafi regime

(a) English Verb (#1902)

einige von Ihnen haben vielleicht davon gehört , dass ich
vor ein paar Wochen eine Anzeige bei Ebay geschaltet habe .

(b) German Article (#590)

Capturing Specific Linguistic Phenomena

CNN Language
Model

- Neurons learning specific word categories such as, law domain

Unit 108: **legal**, **law**, **legislative**

- Better **legal** protection for accident victims.
- These rights are guaranteed under **law**.
- This should be guaranteed by **law**.
- This **legislative** proposal is unusual.
- Animal feed must be safe for animal health.

Unit 711: **should**, **would**, **not**, **can**

- That **would** **not** be democratic.
- That **would** be cheap and it **would** **not** be right.
- This is **not** how it **should** be in a democracy.
- I hope that you **would** **not** want that!
- Europe **cannot** and must **not** tolerate this.

Capturing Specific Linguistic Phenomena

CNN Language
Model

- Neurons learning phrase-level concepts

Phrase

Layer 06, Unit 396: **of this communication** **will**
communication

- That is not the subject **of this communication**.
- That is the purpose **of this communication**.
- I would like to ask the Commissioner for a reply.
- This is impossible without increasing efficiency.
- **Will** we be able to achieve this, Commissioner?

Layer03, Unit 244: **very disappointing** **absolute worst place**

- **very disappointing**, ordered a vegetarian entrée,...
- what the hell did i pay for?...
- **the absolute worst place** i have ever done business with!
- the is by far the worst restaurant i have ever been to...
- this place is a rip off!...

Salient Neurons

LSTM Machine
Translation

- A large number of top salient neurons learn **position, punctuation, conjunction** and **parentheses** concepts

They also violate the relevant Security Council resolutions , in particular resolution 2216 (2015) , and are consistent with the Houthis ' total rejection of the said resolution .

International Law (" Hague Conference ") requested

Switch Neuron

LSTM Machine
Translation

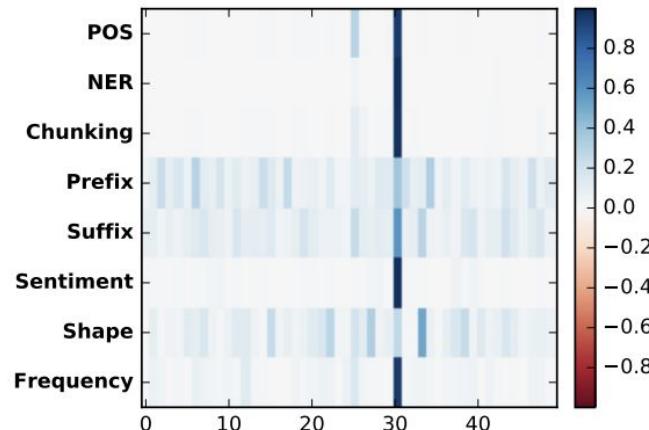
- Tense neurons that toggles between past and present-tense verbs

31 . Recognizes the important contribution of the African Peer Review Mechanism since its inception in improving governance and supporting socioeconomic development in African countries , and recalls in this regard the high-level panel discussion held on 21 October 2013 on Africa 's innovation in governance through 10 years of the African Peer Review Mechanism , organized during the sixty-eighth session of the General Assembly to commemorate the tenth anniversary of the Mechanism ;

Salient Neurons

Static Embeddings

- Two dominant neurons (frequency neurons) found in GloVe



(c) GloVe, no dropout.

Groups of neurons via matrix factorization

DistilGPT2

2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL - HLT 2021) is currently scheduled to take place in Mexico City, Mexico from June 6th to June 11th, 2021. We are monitoring the ongoing global pandemic and will update the conference plans (e.g. moving to a virtual or hybrid format) as needed closer to the conference dates. NAACL - HLT 2021 aims to bring together researchers interested in the design and study of natural language processing technology as well as its applications to new problem areas. With this goal in mind, NAACL - HLT 2021 invites the submission of long and short papers on creative, substantial and unpublished research in all aspects of computational linguistics. More details will be available on the conference website. NAACL - HLT 2021 has a goal of a diverse technical program – in addition to traditional research results, papers may present negative findings, survey an area, announce the creation of a new resource, argue a position, report novel linguistic insights derived using existing techniques, and reproduce, or fail to reproduce, previous results. >> The

Groups of neurons via matrix factorization

DistilGPT2

2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL - HLT 2021) is currently scheduled to take place in Mexico City, Mexico from June 6th to June 11th, 2021. We are monitoring the ongoing global pandemic and will update the conference plans (e.g. moving to a virtual or hybrid format) as needed closer to the conference dates. NAACL - HLT 2021 aims to bring together researchers interested in the design and study of natural language processing technology as well as its applications to new problem areas. With this goal in mind, NAACL - HLT 2021 invites the submission of long and short papers on creative, substantial and unpublished research in all aspects of computational linguistics. More details will be available on the conference website. NAACL - HLT 2021 has a goal of a diverse technical program – in addition to traditional research results, papers may present negative findings, survey an area, announce the creation of a new resource, argue a position, report novel linguistic insights derived using existing techniques, and reproduce, or fail to reproduce, previous results. >> The

Groups of neurons via matrix factorization

DistilGPT2

2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL - HLT 2021) is currently scheduled to take place in Mexico City, Mexico from June 6th to June 11th, 2021. We are monitoring the ongoing global pandemic and will update the conference plans (e.g. moving to a virtual or hybrid format) as needed closer to the conference dates. NAACL - HLT 2021 aims to bring together researchers interested in the design and study of natural language processing technology as well as its applications to new problem areas. With this goal in mind, NAACL - HLT 2021 invites the submission of long and short papers on creative, substantial and unpublished research in all aspects of computational linguistics. More details will be available on the conference website. NAACL - HLT 2021 has a goal of a diverse technical program – in addition to traditional research results, papers may present negative findings, survey an area, announce the creation of a new resource, argue a position, report novel linguistic insights derived using existing techniques, and reproduce, or fail to reproduce, previous results. >> The

Groups of neurons via matrix factorization

DistilGPT2

2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2021) is currently scheduled to take place in Mexico City, Mexico from June 6th to June 11th, 2021. We are monitoring the ongoing global pandemic and will update the conference plans (e.g. moving to a virtual or hybrid format) as needed closer to the conference dates. NAACL-HLT 2021 aims to bring together researchers interested in the design and study of natural language processing technology as well as its applications to new problem areas. With this goal in mind, NAACL-HLT 2021 invites the submission of long and short papers on creative, substantial and unpublished research in all aspects of computational linguistics. More details will be available on the conference website. NAACL-HLT 2021 has a goal of a diverse technical program – in addition to traditional research results, papers may present negative findings, survey an area, announce the creation of a new resource, argue a position, report novel linguistic insights derived using existing techniques, and reproduce, or fail to reproduce, previous results. >> The

Groups of neurons via matrix factorization

DistilGPT2

2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL - HLT 2021) is currently scheduled to take place in Mexico City, Mexico from June 6th to June 11th, 2021. We are monitoring the ongoing global pandemic and will update the conference plans (e.g. moving to a virtual or hybrid format) as needed closer to the conference dates. NAACL - HLT 2021 aims to bring together researchers interested in the design and study of natural language processing technology as well as its applications to new problem areas. With this goal in mind, NAACL - HLT 2021 invites the submission of long and short papers on creative, substantial and unpublished research in all aspects of computational linguistics. More details will be available on the conference website. NAACL - HLT 2021 has a goal of a diverse technical program – in addition to traditional research results, papers may present negative findings, survey an area, announce the creation of a new resource, argue a position, report novel linguistic insights derived using existing techniques, and reproduce, or fail to reproduce, previous results. >> The

Top k Contexts that Activate a Neuron

Multi-Modal GRU

Table 1: Contexts from the top 20 trigrams for example hidden units in each pathway.

	VISUAL	TEXTUAL
Electronic items	and a laptop, cables on it, camera parts and, and cables on, cords and cables	other cars driving, engine car traveling, watches cars racing, with passengers driving
Group of people	crowd together on, team run on, group of men, of men behind, team crowd together	a sandy beach, lush green hillside, of a beach, a rocky hillside, a dry river
Salads	with broccoli carrots, with noodles corn, bowl with meat, salad has broccoli, salad with broccoli	tomatoes on a, food on a, broccoli on a, vegetables on a, vegetables on a
	two teddy bears, teddy bears posing, three teddy bears, teddy bears sitting, bears sitting next	tennis court waiting, water and waiting, table and ready, cooked and ready, into a waiting
	a dirt track, a race track, a stunt jump, stunt jump in, in the air	on the shore, on the floor, at the end, at the shore, on the side

Generated inputs unfolds hidden patterns

Multi-Modal GRU

- Generate examples with respect to a neuron using Gumble softmax
- **Neuron 522:**
 - Corpus search reflects only “horse”
 - The generated example shows that the neuron is also activated by motorcycles and is a race neuron

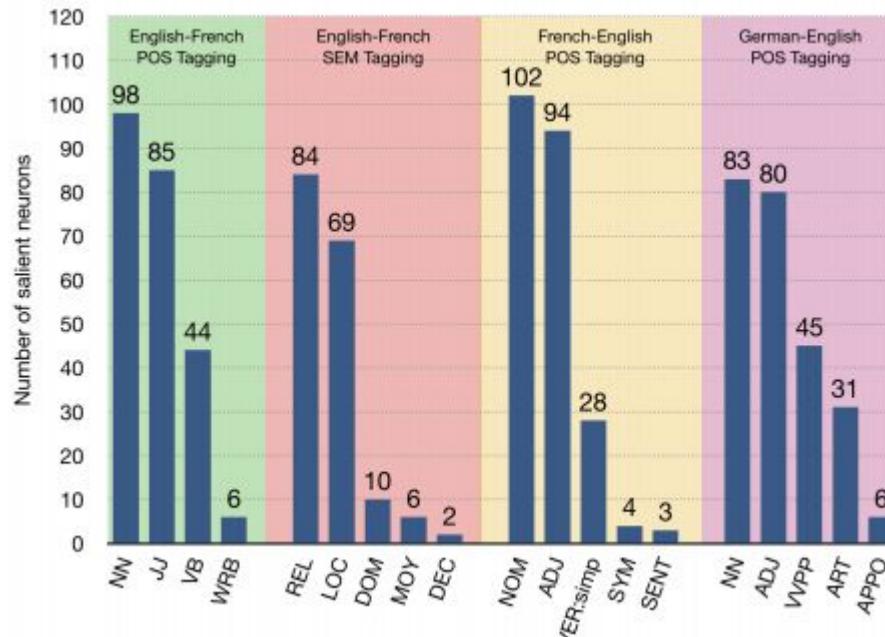
crp	a woman sitting under an	13.28
gbl	campbell lawn raincoat under an	17.54
314th neuron (“umbrella”) in language model projection layer		
crp	finish line at a horse	9.45
gbl	horsed horseback motocycles enthusiast they	13.32
522nd neuron (“race”) in language model projection layer		
crp	the view through a car	10.42
gbl	logging jeep watch through cracked	14.87
957th neuron (“windshield”) in language model projection layer		
crp	a giraffe looks to its	10.64
gbl	fest stares stares to their	13.22
973th neuron (“left”) in language model projection layer		

Table 1: Examples of optimal 5-grams via corpus search and via gradient ascent with gumbel softmax. Spelling errors stem from the Imagenet dictionary.

Localized vs. Distributed Concepts

LSTM Machine Translation

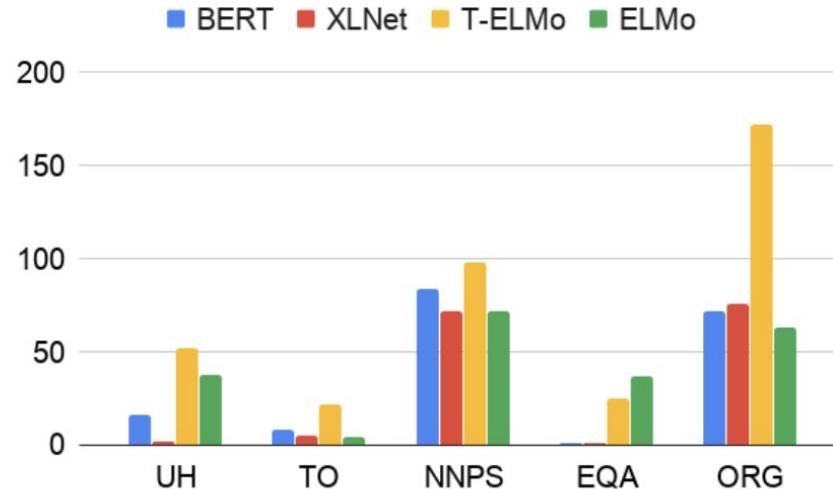
- Linguistic concepts are learned in both localized and distributed fashion



Localized vs. Distributed Concepts

Transformer
Language Models

- Some concepts are localized to fewer neurons
- Consistent across various architectures



Findings and Applications

Concept Discovery
Architectural Analysis
Applications

Minimum set of neurons

Transformer
Language Models

- Small subsets of neurons exist to predict specific linguistic tasks

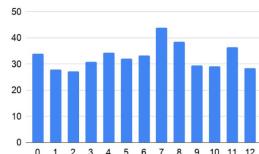
BERT (9984 Neurons)

Task	Number of neurons matching oracle accuracy
POS	4% (399 neurons)
SEM	4% (399 neurons)
Chunking	10% (998 neurons)
CCG Tagging	15% (1497 neurons)

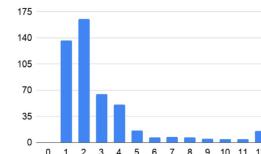
Distribution of neurons across layers

Transformer
Language Models

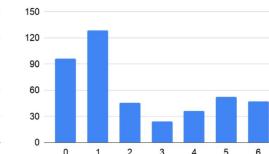
- Neurons that capture basic linguistic information are found at the lower layers
- Those capturing complex syntactic phenomenon were found at the higher layers



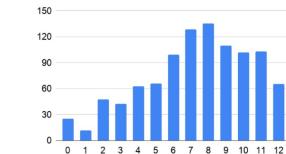
(a) POS – BERT



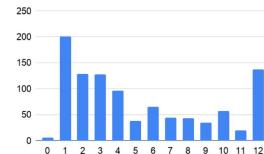
(b) POS – XLNet



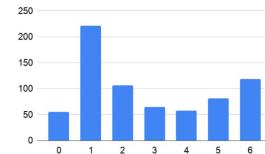
(c) POS – T-ELMo



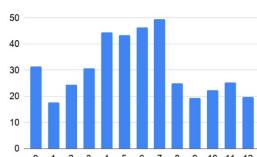
(i) Chunking – BERT



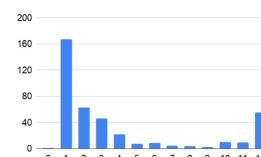
(j) Chunking – XLNet



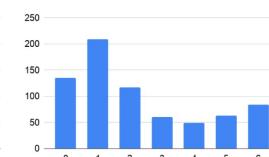
(k) Chunking – T-ELMo



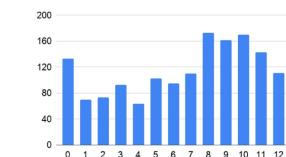
(e) SEM – BERT



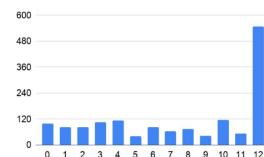
(f) SEM – XLNet



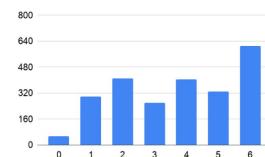
(g) SEM – T-ELMo



(m) CCG – BERT



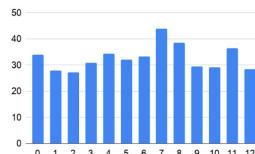
(n) CCG – XLNet



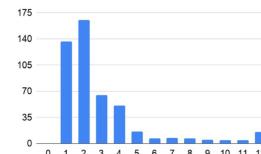
(o) CCG – T-ELMo

Distribution of neurons across layers

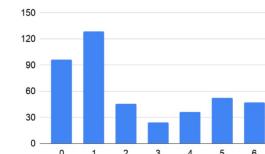
- Neurons that capture basic linguistic information are found at the lower layers
- Those capturing complex syntactic phenomenon were found at the higher layers



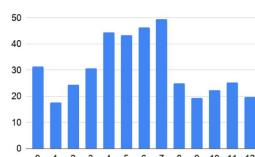
(a) POS – BERT



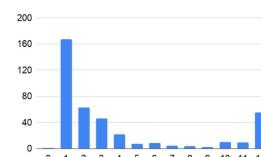
(b) POS – XLNet



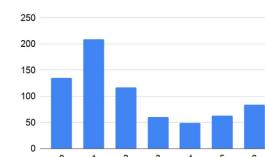
(c) POS – T-ELMo



(e) SEM – BERT



(f) SEM – XLNet

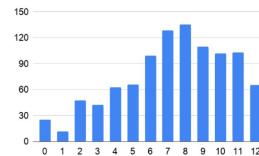


(g) SEM – T-ELMo

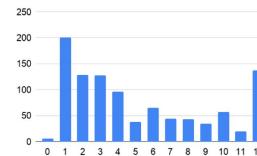
Distribution of neurons across layers

Transformer
Language Models

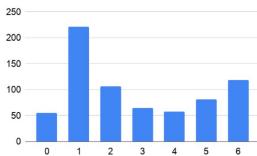
- Neurons that capture basic linguistic information are found at the lower layers
- Those capturing complex syntactic phenomenon were found at the higher layers



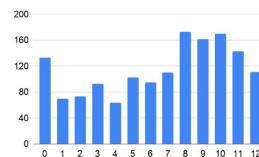
(i) Chunking – BERT



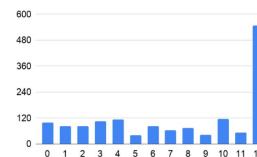
(j) Chunking – XLNet



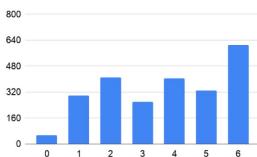
(k) Chunking – T-ELMo



(m) CCG – BERT



(n) CCG – XLNet

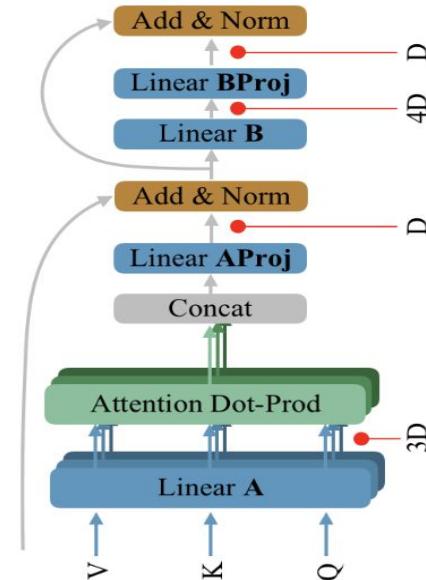


(o) CCG – T-ELMo

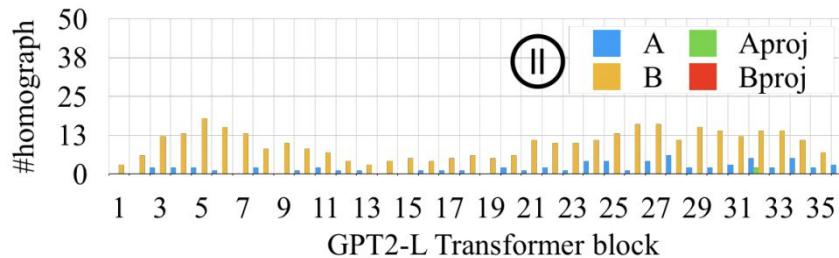
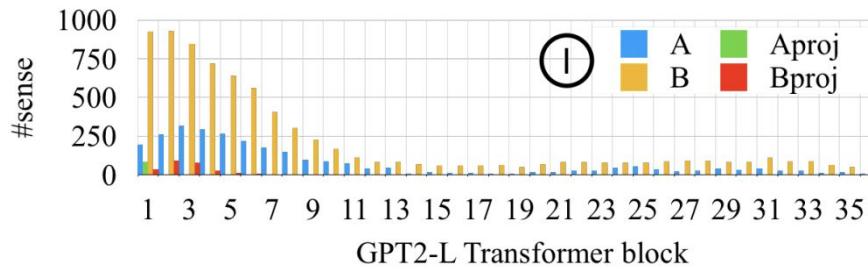
Distribution of neurons within layer block

Transformer
Language Models

- Inside an encoder transformer block, neurons of deeper layers (B) encode more “sense concepts” than earlier layers (A)



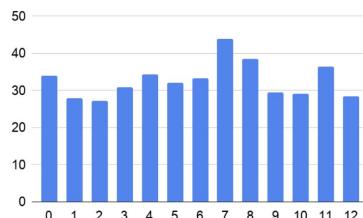
Distribution of neurons within layer block



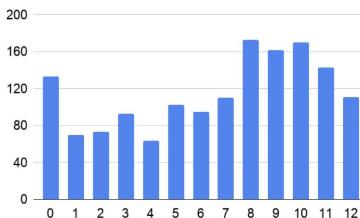
Comparing Architectures

Transformer
Language Models

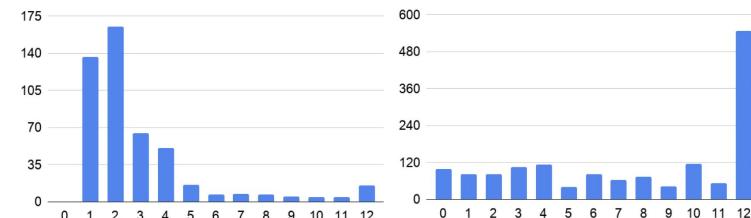
- Information is more distributed in BERT compared to XLNet



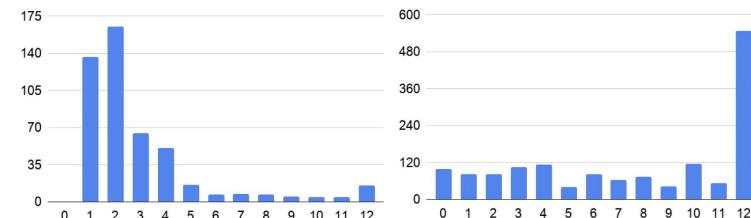
(a) POS – BERT



(m) CCG – BERT

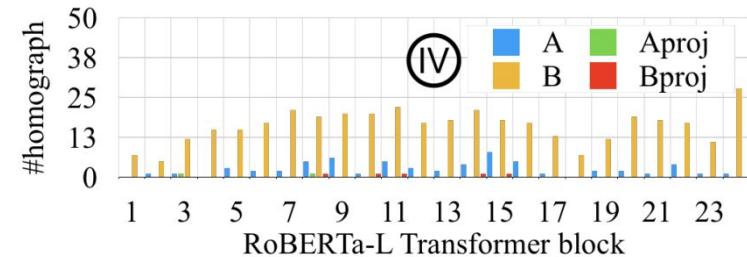
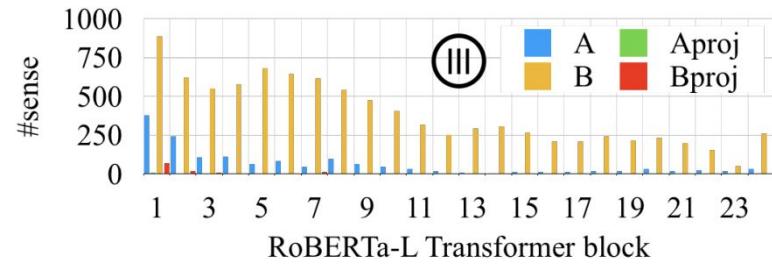
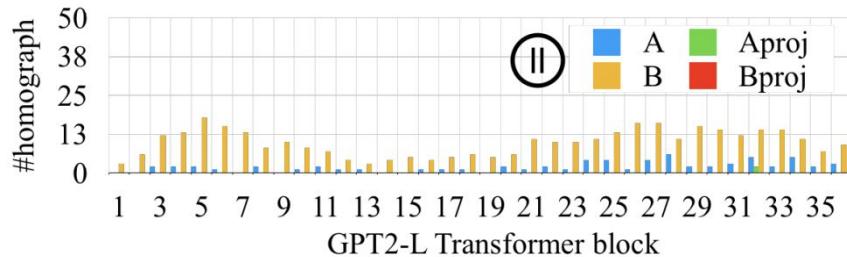
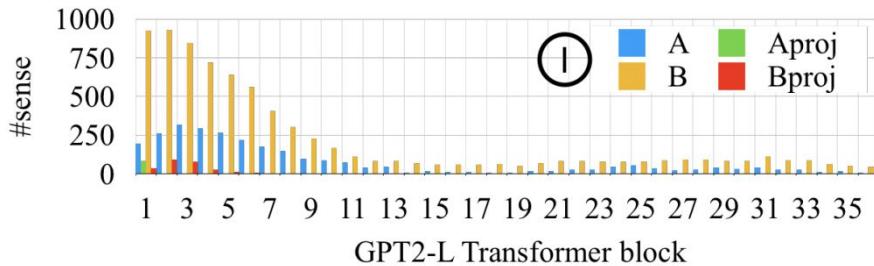


(b) POS – XLNet



(n) CCG – XLNet

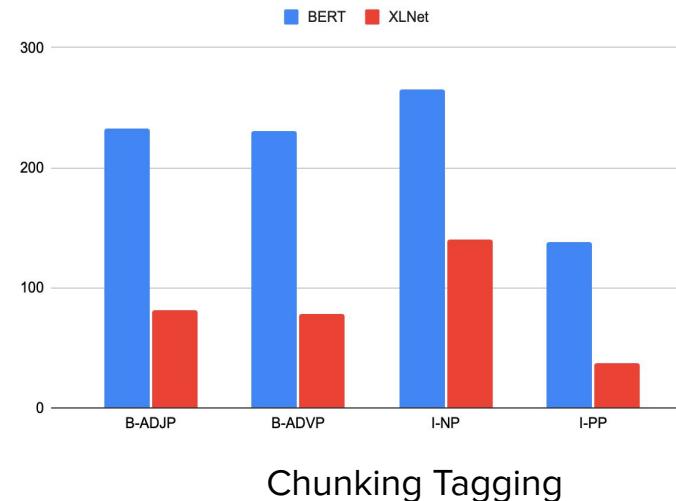
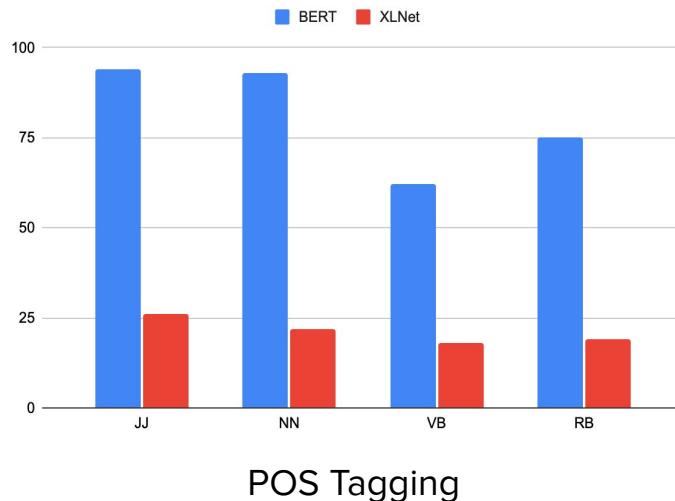
Comparing Architectures



Comparing Architectures

Transformer
Language Models

- XLNet learns linguistic concepts in fewer neurons



Comparing Architectures

Static vs Contextual Embeddings

- fastText requires fewer neurons than BERT to capture morpho-syntactic concepts

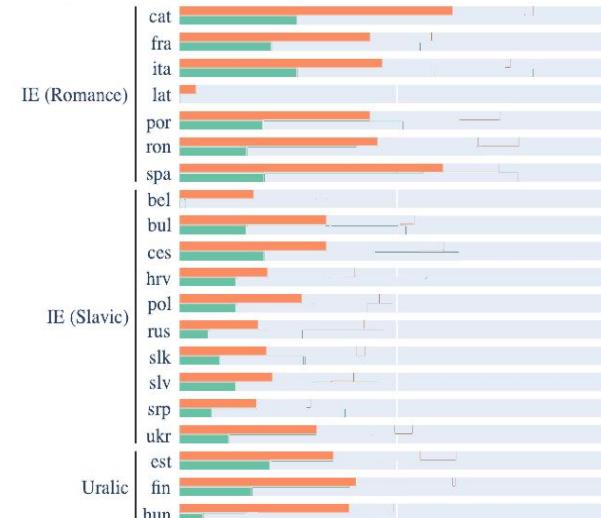
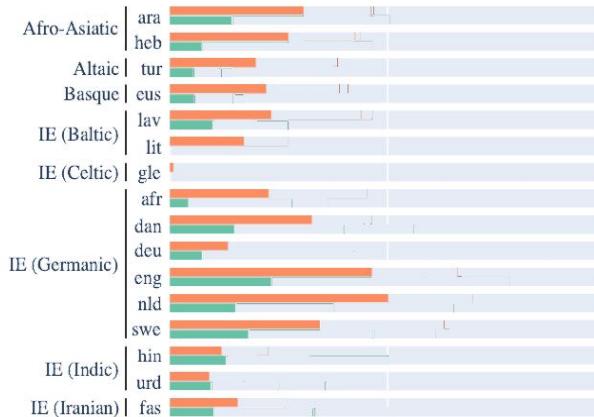


Figure modified for emphasis to show performance using 2 dimensions only

[Intrinsic Probing through Dimension Selection](#) (Hennigen et al. 2020)

Comparing Architectures

Transformer
Language Models

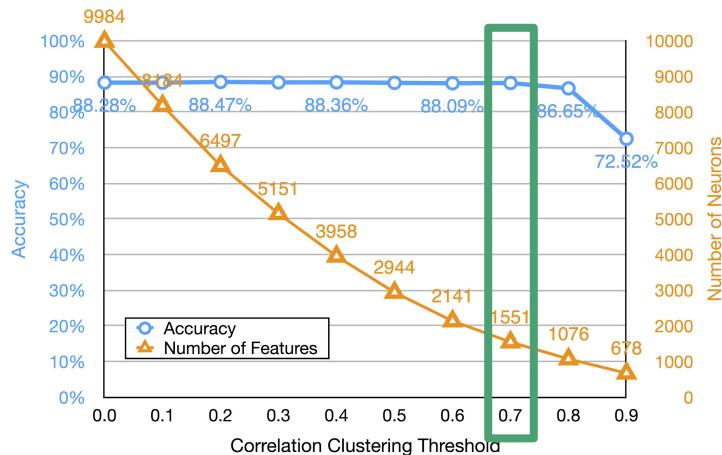
- RoBERTa and GPT have more “specialized” neurons w.r.t. some concepts compared to BERT and DistilBERT

Model	Model size	χ_{γ^*}	
BERT-L	330M	7.19%	
Distilbert	66M	4.02%	χ_{γ^*} shows the percentage of expert (specialized) neurons
GPT2-L	774M	12.92%	
RoBERTa-Lm	355M	15.36%	

Redundancy Analysis

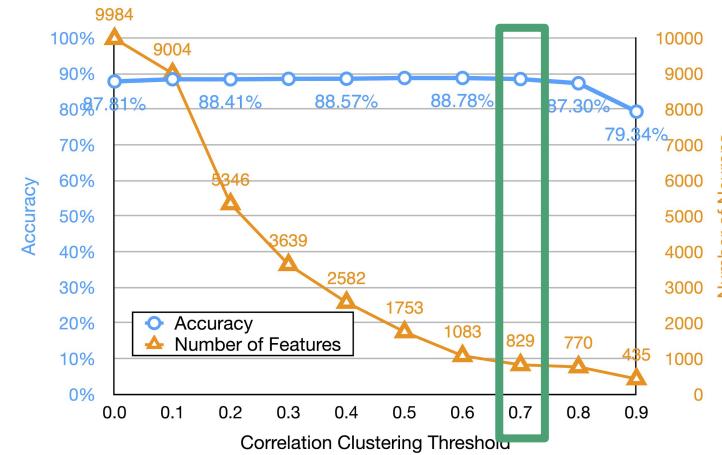
Transformer Language Models

- Substantial amount of neurons are redundant when used as features for transfer learning
 - 85% (BERT) and 92% (XLNet)



BERT

[Analyzing Redundancy in Pretrained Transformer Models](#) (Dalvi et al. 2020)



XLNet

Redundancy Analysis

LSTM NER Model

- Some neurons are exclusive to a single label and some are polysemous
- Information is redundantly available
 - Ablating concept-specific neurons does not depreciate model's overall accuracy

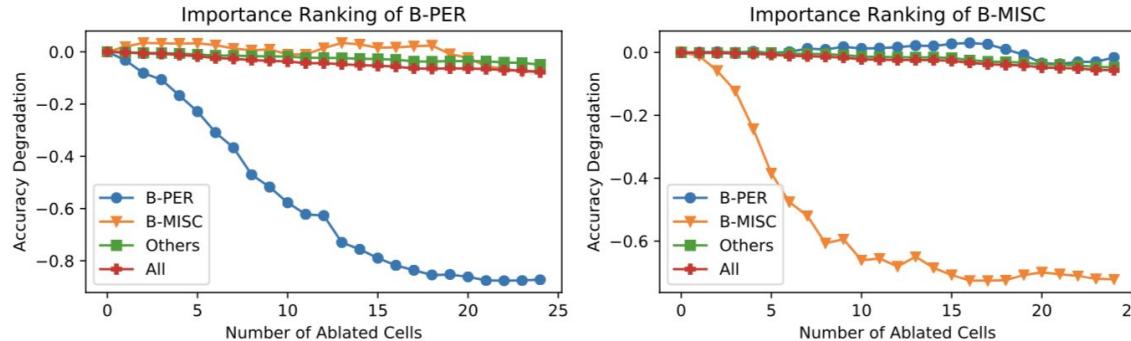
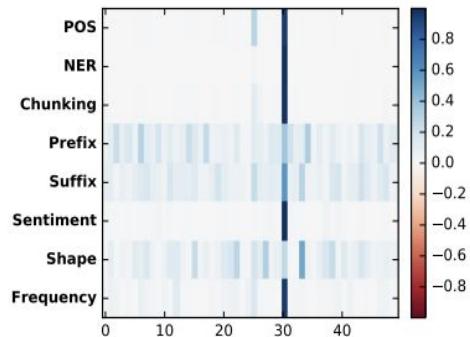


Figure 4: Ablation according to importance ranking of B-PER and B-MISC (yellow columns in Figure 3).

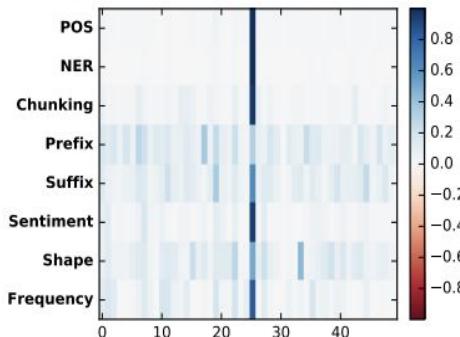
Redundancy Analysis

Static Embeddings

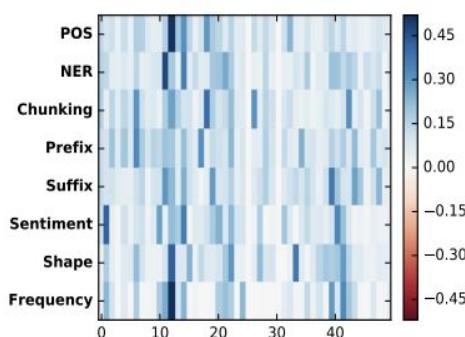
- Two dominant neurons (frequency neurons) found in GloVe



(c) GloVe, no dropout.



(d) GloVe, no dropout; 31rd dimension removed.

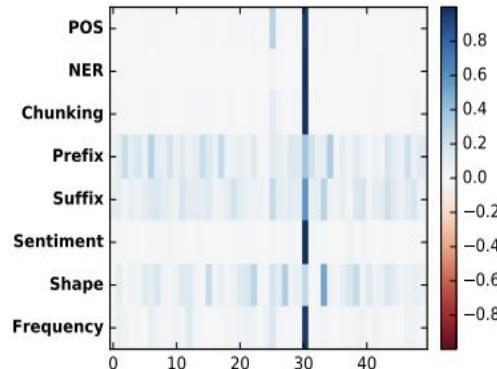


(e) GloVe, no dropout; 31rd, 26th dimensions removed.

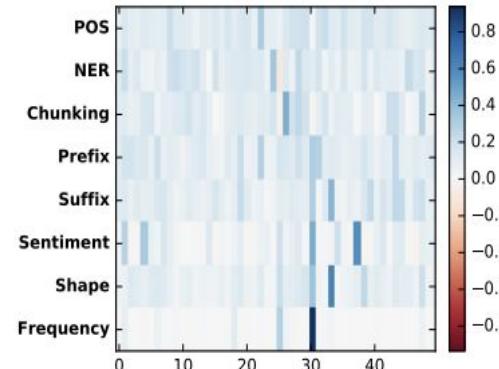
Redundancy Analysis

Static Embeddings

- Information is distributed across the network



(c) GloVe, no dropout.



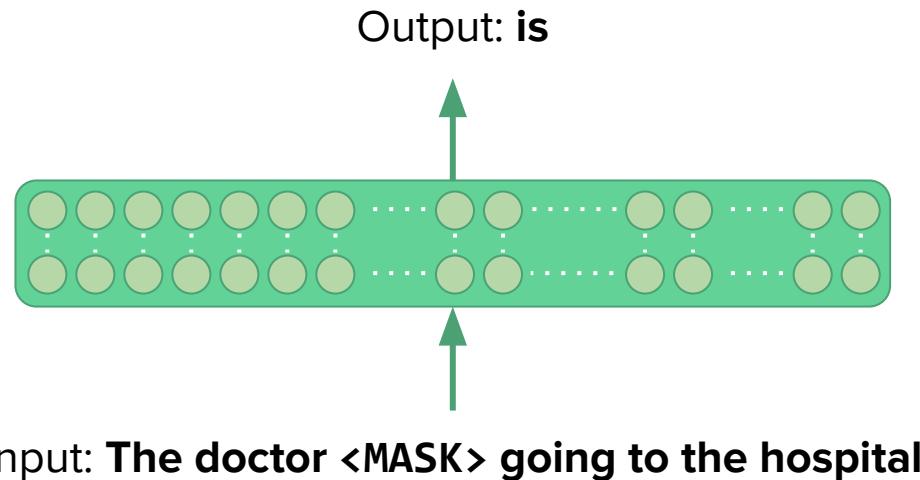
(f) GloVe, with dropout.

Findings and Applications

Concept Discovery
Architectural Analysis
Applications

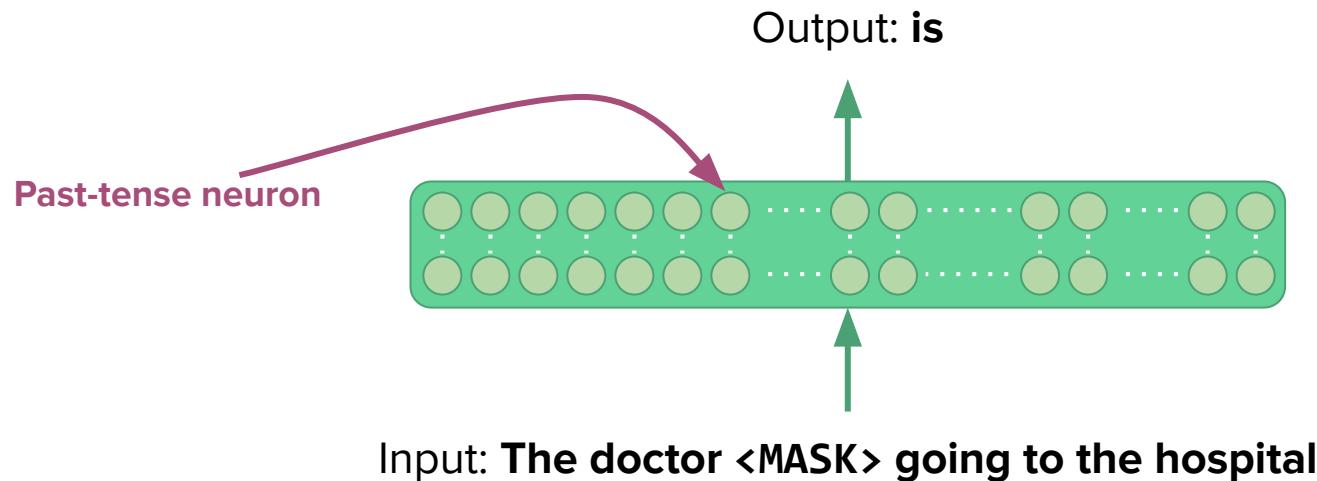
Can we use neurons to manipulate predictions?

- Identifying and changing the activation value of particular neurons can help us control the output of a model



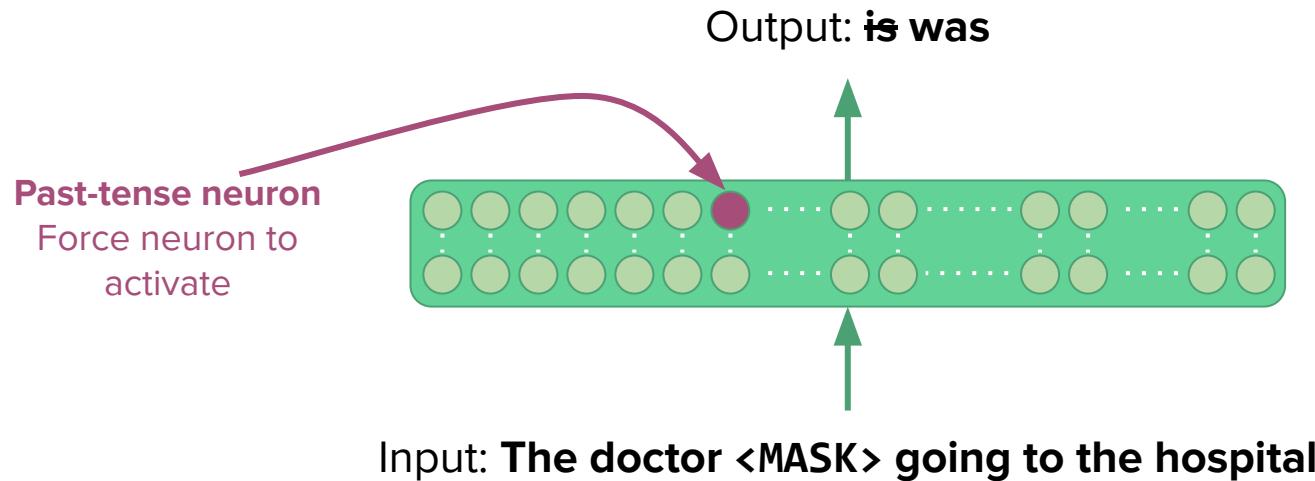
Can we use neurons to manipulate predictions?

- Identifying and changing the activation value of particular neurons can help us control the output of a model



Can we use neurons to manipulate predictions?

- Identifying and changing the activation value of particular neurons can help us control the output of a model



Switch Neuron

- A neuron learning **present and past verb tense** on the opposite spectrum of their activation values

7439th meeting , held on 11 May 2015 .

ISIL itself has published videos depicting people being subjected to a range of abhorrent punishments , including stoning , being pushed-off buildings , decapitation and crucifixion .

UNICEF disbursed emergency cash assistance to tens of thousands of displaced families in camps and UNHCR discontinued cash assistance to vulnerable families which had been internally displaced .

31 . Recognizes the important contribution of the African Peer Review Mechanism since its inception in improving governance and supporting socioeconomic development in African countries , and recalls in this regard the high-level panel discussion held on 21 October 2013 on Africa 's innovation in governance through 10 years of the African Peer Review Mechanism , organized during the sixty-eighth session of the General Assembly to commemorate the tenth anniversary of the Mechanism ;

Spreads between sovereign bonds in Germany and those in other countries were relatively unaffected by political and market uncertainties concerning Greece in late 2014 and early 2015 .

Manipulating Tense

Neural Machine
Translation

- Translation output can be flipped between past and present tense by controlling the activations of the *tense neurons*
- Successful in 67% cases

	Translation	Tense
Arabic	وأيدت\ وتأيد اللجنة {جهود\الجهود التي تبذلها} السلطات	past/present
French	Le Comité <u>a appuyé/appuie</u> les efforts des autorités	past/present
Spanish	El Comité <u>apoyó/apoyaba/apoya</u> los esfuerzos de las autoridades	past/impf./present
Russian	Комитет <u>поддержал/поддерживает</u> усилия властей	past/present
Chinese	委员会 支持 当局 的 努力 / 委员会 <u>正在 支持</u> 当局 的 努力	untensed/present

Controlling tense when translating “The committee *supported* the efforts of the authorities”.

Manipulating Gender

Neural Machine
Translation

- Similar findings were reported for gender and number agreement

Translation	Gen	Translation	Gen
Los partidos interados	ms.	Temas relativos a la información	ms.
Las partes interesadas	fm.	Cuestiones relativas a la información	fm.

Controlling gender when translating “The interested *parties*” (left) and “*Question* relating to information” (right) to Spanish

- Success rate is still quite low (21% for gender, 33% for number)

Inducing Concepts in Text Generation

Transformer
Language Models

- Concepts can be induced to change outputs

K forced	Once upon a time + Generated induced for concept bird%1:05:00 (warm-blooded egg-laying vertebrates)
0 (0%)	, I had a friend who used to teach high school English and he was like, "Oh, all you have to do is just get out there
40 (0.009%)	, many of these treasures were worth hundreds of thousands of dollars.\n But this isn't the first time that a horse has been
60 (0.015%)	, through a freak occurrence, an invasion of house sparrows, which so often reduces the black-browed this nation recreates through
80 (0.019%)	, our own ancestors rode about on chicken-like air wings.\n But this wonder of the air has no such wings.\n Taking down
200 (0.048%)	of year, birds chase each and watching. flot racing form, bird, bird bird bird bird bird bird bird bird bird bird, Bird bird
Once upon a time + Generated induced for concept lead%1:07:02 (an advantage held by a competitor in a race)	
50 (0.012%)	the left-hander would always start at the front in the first two instances, but when Mauricio Gaponi rose to the podium,
Once upon a time + Generated induced for concept lead%1:27:00 (a soft heavy toxic malleable metallic element)	
100 (0.024%)	a crust layer was applied to a partially fortified nickel base, thereby causing to zinc- and copper- ground element cob. The occurrence of those metal and chrome

Open Questions

- Neurons are redundant and manipulating a particular neuron may cause the network to fallback to other neurons designated for the job
- It is non-trivial how to
 - Identify which group of neurons play part in effectively manipulating the output
 - Set an optimal value for a neuron at which it can effectively manipulate the output
 - Deal with the polysemous neurons that play varied roles in the network
- Neurons are interconnected and manipulating a neuron may affect other neurons resulting in unstable network performance

Efficient Feature-based Transfer Learning

Transformer
Language Models

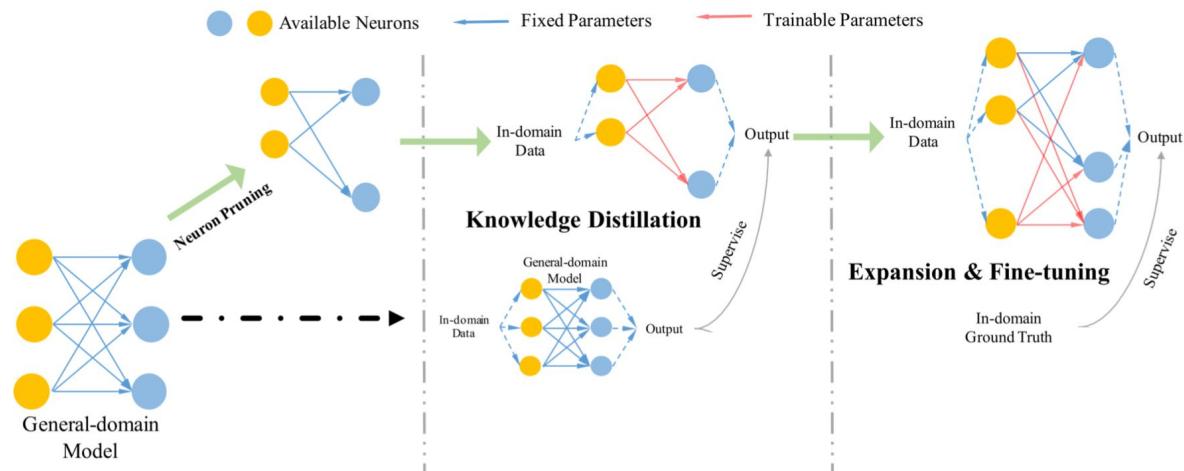
- Analyzing redundancy can be useful for efficient **feature-based** transfer learning
- Combine correlation clustering with Neuron ranking and select the most salient neurons
- Number of features required for transfer learning can be reduced significantly without loss in performance

	Sequence Classification		Sequence Labeling	
	BERT	XLNet	BERT	XLNet
Oracle	93.0%	93.4%	85.5%	84.8%
Neurons			9984	
CCFS	92.0%	92.2%	84.0%	84.0%
Neurons	425	400	90	150
% Reduct.	95.7%↓	96.0%↓	99.0%↓	98.5%↓

Domain Adaptation using Neuron Pruning

Transformer MT
Models

- Prune the network based on important neurons
- Restore the pruned network's performance using student-teaching learning
- Expand the network to the original size and fine-tune additional parameters on in-domain data



Generating compositional explanations

BiLSTM NLI Model

- Each concept defines an explanation of a neuron
- A compositional explanation can be generated using logical operators
- For example: several neurons are gender sensitive and activate for **contradiction** when the premise, but not the hypothesis, contains the word “man” in an NLI task

Unit 870 (gender-sensitive)

((((NOT hyp:man) AND pre:man) OR hyp:eating)
AND (NOT pre:woman)) OR hyp:dancing
IoU **0.123** w_{entail} **-0.046** w_{neutral} **-0.021** w_{contra} **0.040**

Pre A guy pointing at a giant blackberry.

Hyp A woman tearing down a giant display.

Act **29.31** True **contra** Pred **contra**

Pre A man in a hat is working with...flowers.

Hyp Women are working with flowers.

Act **27.64** True **contra** Pred **contra**

Adversarial examples

BiLSTM NLI Model

- Spurious correlations (“nobody” in hypothesis)
- Create adversarial examples by looking at such explanations

Unit 39 (nobody in hypothesis)

hyp:nobody AND (NOT pre:hair) AND (NOT pre:RB) AND (NOT pre:’s)

IoU **0.465** w_{entail} **-0.117** w_{neutral} **-0.053** w_{contra} **0.047**

Pre Three women prepare a meal in a kitchen.

Orig Hyp The ladies are cooking.

Adv Hyp **Nobodybut** the ladies are cooking.

True entail $\xrightarrow{\text{adv}}$ neutral Pred entail $\xrightarrow{\text{adv}}$ contra

Adversarial examples

BiLSTM NLI Model

- Spurious correlations (high overlap between premise and hypothesis)
- Create adversarial examples by looking at such explanations

Unit 99 (high overlap)

((NOT hyp:JJ) AND overlap-75% AND (NOT pre:people)) OR pre:basket OR pre:tv
IoU **0.118** w_{entail} **0.043** w_{neutral} **-0.029** w_{contra} **-0.021**

Pre A woman in a light blue jacket is riding a bike.

Hyp A women in a jacket riding a bike.

Act **19.13** True **entail** Pred **entail**

Pre A girl in a pumpkin dress sitting at a table.

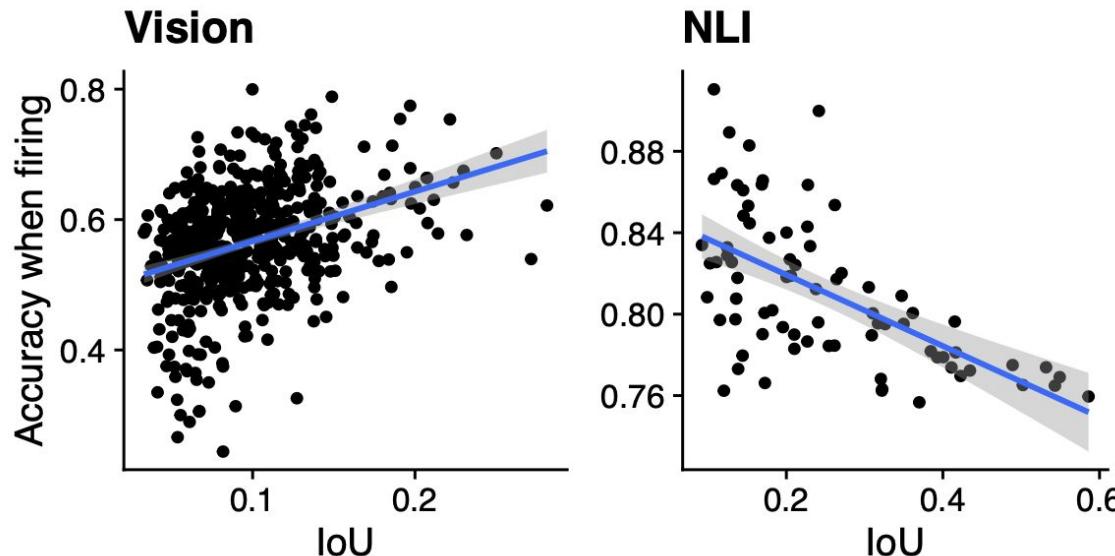
Hyp There is a girl in a pumpkin dress sitting at a table.

Act **17.84** True **entail** Pred **entail**

Connecting Interpretation and Performance

BiLSTM NLI Model

- Neurons that are more explainable correlate with prediction accuracy



Connecting Interpretation and Performance

Neural Machine
Translation

- Ablating neurons by their rank gives us an idea of their importance for the prediction
- Neurons that capture morphology and semantics are not the most salient neurons for the translation task
- Semantics is slightly more important than morphology

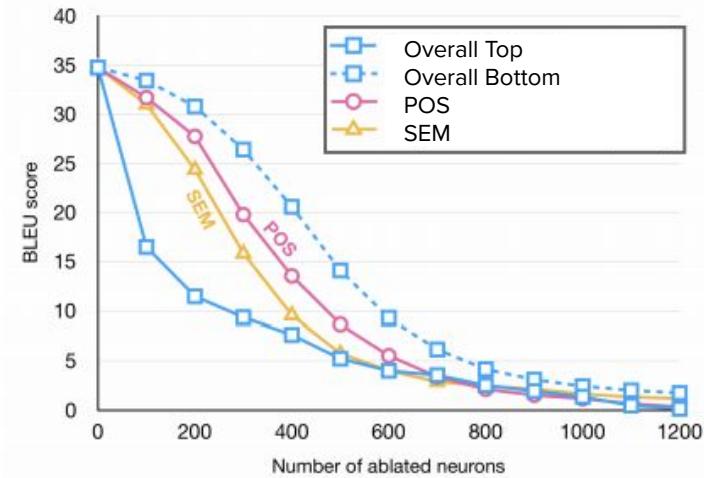


Figure 8: Effect on translation when ablating neurons in the order determined by both methods on the EN-FR model

Concept Analysis

Datasets

What kinds of datasets can be used for neuron probing?

- Properties may be encoded in a single neuron or across many neurons
- Techniques vs datasets
 - Not all methods are adept at finding “distributed properties” across neurons
- Granularity of a concept: If a concept is too diverse, look at sub concepts
- Auto-generation
 - Sentence length
 - Word presence/absence
 - Parse trees
 - WordNet/SenseNet

What kinds of datasets have been used?

- Penn Treebank part of speech tags ([Compositional Explanations of Neurons](#))
- Semantic Tags Groningen Parallel Meaning Bank ([What Is One Grain of Sand in the Desert?](#))
- Syntactic Tags Universal Dependencies data set ([What Is One Grain of Sand in the Desert?](#))
- CCG Super tags English CCGBank ([What Is One Grain of Sand in the Desert?](#))
- OneSec dataset: Sentences -> keyword+sense ([Finding Experts in Transformer Models](#))
- UD 1.2 ([Analyzing Linguistic Knowledge in Sequential Model of Sentence](#))
- UD v2.1 treebanks, 36 languages ([Intrinsic Probing through Dimension Selection](#))
- Number-Agreement set ([The Emergence of Number and Syntax Units in LSTM Language Models](#))
- Syntactic Depth Data-Set ([The Emergence of Number and Syntax Units in LSTM Language Models](#))
- CoNLL2003 Named Entity recognition dataset ([What Part of the Neural Network Does This? Understanding LSTMs by Measuring and Dissecting Neurons](#))
- Semantic Roles PropBank/Proto-Roles ([Asking without Telling: Exploring Latent Ontologies in Contextual Representations](#))
- Autogenerated: words + morphemes + phrases from constituency-based parse tree ([Discovery Of Natural Language Concepts In Individual Units Of CNNs](#))
- Most frequent words ([Compositional Explanations of Neurons](#))
- Stanford Sentiment Treebank ([Visualizing and Understanding Neural Models in NLP](#))
- Simple properties: Length, full stop, quotes, parenthesis

Concept Analysis

Practical

Practical toolkits

- [LSTMVis](#): Recurrent neural networks analysis
- [Ecco](#): Interactive analysis for pre-trained language models
- [NeuroX](#): Neuron analysis toolkit for deep NLP models

Practical toolkits: LSTMVis



Practical toolkits: Ecco

2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL - HLT 2021) is currently scheduled to take place in Mexico City, Mexico from June 6th to June 11th, 2021. We are monitoring the ongoing global pandemic and will update the conference plans (e.g. moving to a virtual or hybrid format) as needed closer to the conference dates. NAACL - HLT 2021 aims to bring together researchers interested in the design and study of natural language processing technology as well as its applications to new problem areas. With this goal in mind, NAACL - HLT 2021 invites the submission of long and short papers on creative, substantial and unpublished research in all aspects of computational linguistics. More details will be available on the conference website. NAACL - HLT 2021 has a goal of a diverse technical program – in addition to traditional research results, papers may present negative findings, survey an area, announce the creation of a new resource, argue a position, report novel linguistic insights derived using existing techniques, and reproduce, or fail to reproduce, previous results. >> [The](#)

Practical toolkits: NeuroX

- NeuroX is a python toolkit with several neuron analysis techniques and probes implemented
- Supports neuron activation extraction from all [transformers](#) models, more libraries coming soon!
- Provides methods to extract important neurons from a model given a concept
- Provides methods to perform clustering analysis on top of neuron activations, like correlation clustering
- Includes helper functions to load and process data, as well as visualize individual neuron activations

Practical toolkits: NeuroX

- Let's dive deeper into the toolkit and see how we can use the library.
- All of the following code and data files can be found at [NeuroX/examples/End to End Example.ipynb](#).

NeuroX: Extracting activations

- Before performing any analysis, we first need to have activations on some given text.

```
!cat test.in
```

```
This is a test sentence .
This is another sentence .
The NeuroX toolkit can be used for neuron analysis !
This sentence contains an <<UNKNOWN>> word .
Individual neuron activations can be insightful .
It is a beautiful day outside .
The red car zoomed by quickly !
Is this a question ?
There are more than 10000 neurons in some models .
Hope you are enjoying the tutorial !
```

NeuroX: Extracting activations

- Extraction is as simple as calling the appropriate extractor (in our case, let's consider a transformers model):

```
import neurox.data.extraction.transformers_extractor as transformers_extractor
transformers_extractor.extract_representations('bert-base-uncased',
    'test.in',
    'activations.json',
    aggregation="average" #last, first
)
```

- This creates a file called activations.json with all the activations associated with all the tokens in test.in

NeuroX: Loading data

- We can now load supervised data for some concepts and the activations to perform further analysis.
- Consider the following test.label file that has word-level annotations for every token in test.in

```
!cat test.in
```

This is a test sentence .
This is another sentence .
The NeuroX toolkit can be used for neuron analysis !
This sentence contains an <>UNKNOWN<> word .
Individual neuron activations can be insightful .
It is a beautiful day outside .
The red car zoomed by quickly !
Is this a question ?
There are more than 10000 neurons in some models .
Hope you are enjoying the tutorial !

```
!cat test.label
```

DT VBZ DT NN NN .
DT VBZ DT NN .
DT NNP NN MD VB VBN IN JJ NN .
DT NN VBZ DT NN NN .
NNP CC NNS MD VB JJ .
PRP VBZ DT JJ NN IN .
DT JJ NN VBN IN RB .
VBZ DT DT NN .
EX VBP JJR IN CD NNS IN DT NNS .
NN PRP VBP VBG DT NN .

NeuroX: Loading data

- First let us load the extracted activations

```
import neurox.data.loader as data_loader
activations, num_layers = data_loader.load_activations('activations.json', 768)
```

Loading json activations from activations.json...

```
print(activations)
```

```
[array([[ -0.6485135 ,  0.67392403, -0.09324985, ..., -0.45096967,
        0.46059093,  0.93923897],
       [-0.62703037, -0.06331295, -0.31427881, ..., -0.28688025,
        0.12916157,  1.09747875],
       [ 0.39637804,  0.31565616,  0.02008923, ..., -0.18841645,
        0.18347587,  1.51162875],
       [ 0.60103226, -0.69701707, -0.2000732 , ..., -0.51888162,
        -0.24854559,  0.26434106],
       [-0.24404168, -0.66105956, -0.27814269, ..., -0.15310135,
        -0.08717065,  0.15071221], array([-0.36680133,  0.361505,      0.22041463,      -0.
```

NeuroX: Loading data

- Next, we can load the supervised data

```
# load_data also does sanity checks for parallelism between tokens, labels and activations
tokens = data_loader.load_data('test.in',
                               'test.label',
                               activations,
                               512 # max_sent_l
                               )
```

- tokens and activations will now contain parallel source tokens, target labels and neuron activations

NeuroX: Neuron Probing

- We can now perform Neuron Probing using a linear probe
- The method `create_tensors` from `neurox.interpretation.utils` can preprocess the data into tensors, which can then be fed into the training and evaluation pipeline

```
import neurox.interpretation.utils as utils
X, y, mapping = utils.create_tensors(tokens, activations, 'NN')
label2idx, idx2label, src2idx, idx2src = mapping
```

```
Number of tokens: 71
length of source dictionary: 51
length of target dictionary: 19
71
Total instances: 71
['for', 'the', 'outside', 'activations', 'red', 'test', '10000', 'enjoying', 'It', 'The', 'This', '<<UNKNOWN>>', 'tutorial', 'question', 'than', 'neuron', 'is', 'you', 'be', 'Hope']
```

NeuroX: Neuron Probing

- Next, we can train a linear probe for our classification task

```
import neurox.interpretation.linear_probe as linear_probe  
probe = linear_probe.train_logistic_regression_probe(X, y, lambda_l1=0.001, lambda_l2=0.001)
```

```
Training classification probe  
Creating model...  
Number of training instances: 71  
Number of classes: 19
```

```
epoch [1/10]:  3/? [00:00<00:00, 75.10it/s]
```

```
Epoch: [1/10], Loss: 0.1456
```

```
epoch [2/10]:  3/? [00:00<00:00, 77.77it/s]
```

```
Epoch: [2/10], Loss: 0.0448
```

NeuroX: Neuron Probing

- The library also has methods to evaluate a trained probe on various datasets using various metrics

```
linear_probe.evaluate_probe(probe, X, y, idx_to_class=idx2label)
```

Evaluating:  3/? [00:00<00:00, 75.76it/s]

Score (accuracy) of the probe: **1.00**

```
{'__OVERALL__': 1.0,
 'IN': 1.0,
 'NNP': 1.0,
 'PRP': 1.0,
 'JJR': 1.0,
 'JJ': 1.0,
 'EX': 1.0,
 'CC': 1.0,
 '.'': 1.0.
```

NeuroX: Neuron Probing

- Given a well trained probe, we can get the top neurons for every class, as well as the overall top neurons

```
top_neurons, top_neurons_per_class = linear_probe.get_top_neurons(probe, 0.01, label2idx)
print(top_neurons)
```

```
[3584 2563 3588    6 6672 2072 4637 5662 3102 4138 8235 3115 1587 1081
 2108 2621  577 5697 3142 5703 5210 3674 8292 5233   126 2687 6792 7820
 5778 8856 5788 1180 2217 4778 9910 5816   188 1726 5824 8897 1732 9925
 5316 3271  205 5838   725 7894 5336 8932 6886 4854 2812   767 768 4355
 5895 7983 1329 2866 1342 9539 8008 8012 8017 4946 1365 5977 6492 7519
 7523 7524  358 2416 4464 4466 4979 9083 1925 2438 7566 9616 4502 1948
 1439 2474  941 3516 5052 2501 4041 9683 5588 6102 5089 7653 4582 5618
 8183 3068]
```

NeuroX: Neuron Probing

- The neurox.interpretation.ablation module provides several methods to filter activations by either masking certain neurons or removing them completely

```
X_zeroed = ablation.zero_out_activations_keep_neurons(X, top_neurons)
linear_probe.evaluate_probe(probe, X_zeroed, y, idx_to_class=idx2label)
```

Evaluating:  3? [00:00<00:00, 87.27it/s]

Score (accuracy) of the probe: 0.97

```
{'__OVERALL__': 0.971830985915493,
 'IN': 1.0,
 'NNP': 1.0,
 'PRP': 1.0,
 'NNR': 1.0}
```

- Masking can help in evaluating the probe by ablation, while removing neurons can help in understanding the function of specific sets of neurons.

NeuroX: Neuron Probing

- We can also extract a global ordering of neurons, from most to least important

```
ordering, cutoffs = linear_probe.get_neuron_ordering(probe, label2idx)
```

100%  101/101 [00:00<00:00, 153.14it/s]

```
ordering[:10]
```

```
[5336, 4041, 5788, 3584, 768, 2563, 3588, 4355, 6, 5895]
```

NeuroX: Neuron Probing

- It is also a common practice to train probes on certain subsets of neurons (say top N from global ordering, or top neurons for class 1 etc).
- Similarly, one can also train a probe for a single layer to limit their analysis

```
layer_0_X = ablation.filter_activations_by_layers(X, [0], 13)
print(layer_0_X.shape)
probe_layer_0 = linear_probe.train_logistic_regression_probe(layer_0_X, y, lambda_l1=0.001,
linear_probe.evaluate_probe(probe_layer_0, layer_0_X, y, idx_to_class=idx2label)
```

```
(71, 768)
Training classification probe
Creating model...
Number of training instances: 71
Number of classes: 19

epoch [1/10]:  3/? [00:00<00:00, 63.50it/s]
```

NeuroX: Visualizing Activations

- Finally, the library also provides some helper methods to visualize individual neuron activations to qualitatively confirm or analyze them

```
import neurox.analysis.visualization as visualization

visualization.visualize_activations(
    ".join(tokens['source'][9]),
    activations[9][:, 2]
)
```

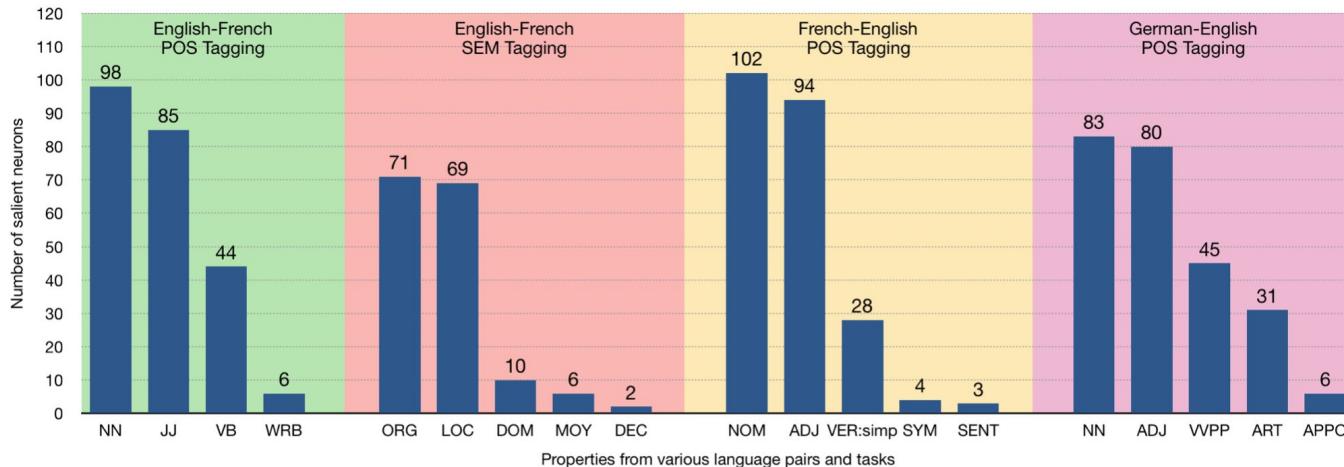
Hope you are enjoying the tutorial !

NeuroX: Example usage

- Let us take a quick look at a few examples of successful analyses done using this toolkit

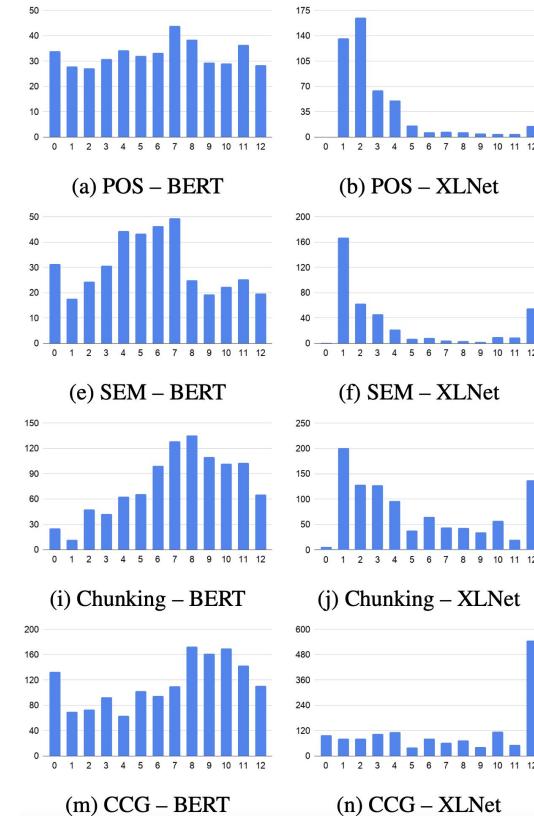
NeuroX: Example usage

- The `neurox.interpretation.linear_probe.get_top_neurons` method can be used to extract top neurons for individual classes
- The number of top neurons per class can help us analyze how concepts are being represented in a network



NeuroX: Example usage

- The `neurox.interpretation.ablation` module can be used to perform layer wise analysis
- We can use the overall accuracy of each layer to hypothesize about the role of the layers
- We can also see how the overall top neurons are spread across different layers in different models



NeuroX

- Visit <https://github.com/fdalvi/NeuroX/> to start using the library
- The API documentation is available at <https://neurox.qcri.org/docs/>
- A GUI application that uses the NeuroX toolkit is available at
<https://neurox.qcri.org/demo/>
- Feel free to open issues on the Github repo for bugs (and help us improve our test suite) as well as feature requests!

Attribution Analysis

What's missing!

- Concept analysis identifies neurons **with respect to a concept**
 - What concepts are captured by the neurons?
 - Where these concepts are learned in the network?
- However, it is not clear if these concept neurons are used in any prediction

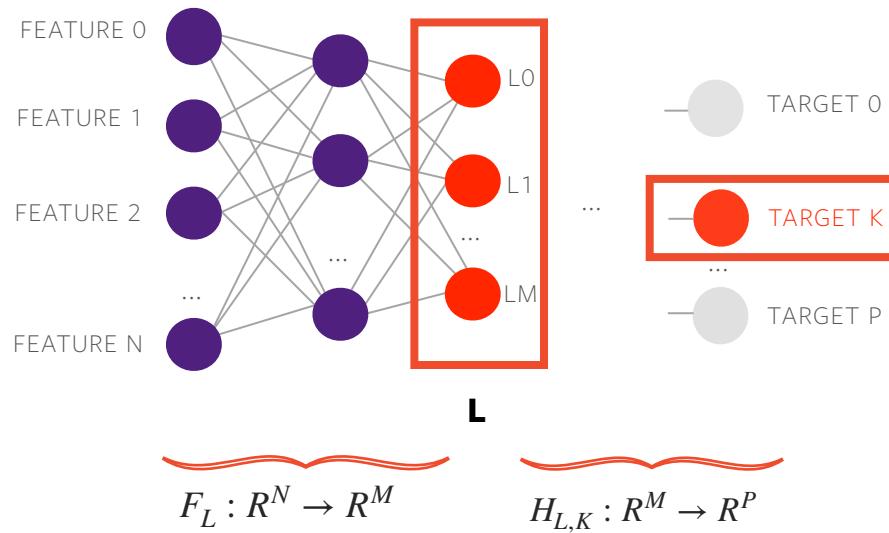
Another line of research called **Attribution Analysis** looks at the contribution of a neuron in a specific prediction irrespective of a concept



Causation analysis using neuron attribution techniques for a specific prediction



ATTRIBUTING TO A HIDDEN LAYER OF A NEURAL NETWORK

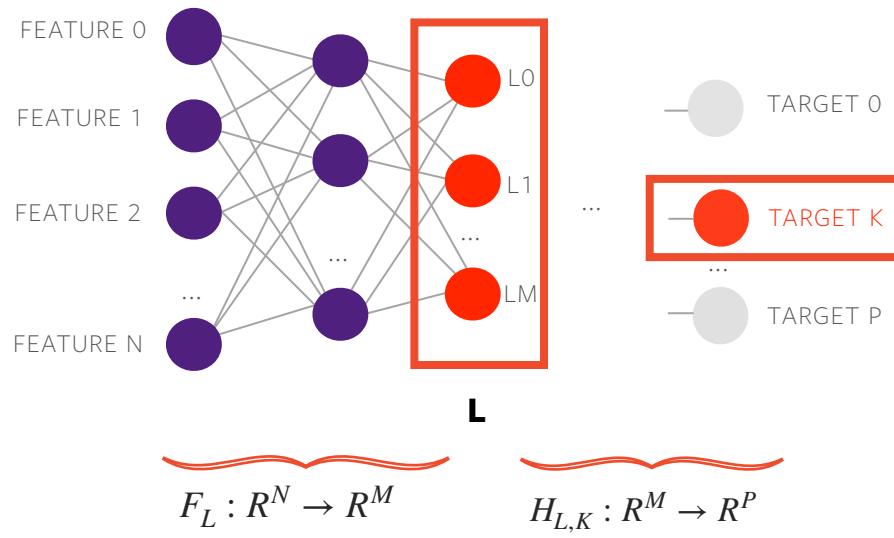


$F_C : R^N \rightarrow R^P$ - a NN Function

C - the prediction class



SALIENCY



$F_C : R^N \rightarrow R^P$ - a NN Function

C - the prediction class

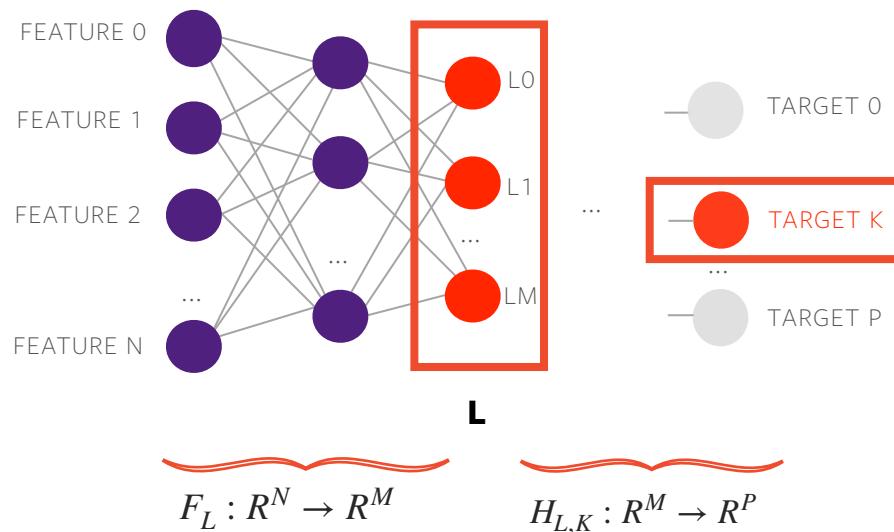
- Infinitesimal change in the neuron activations that changes the output

- $\Phi_c(F_L(x)) = \frac{\partial F_c(x)}{\partial F_L(x)}^*$, where $x \in \mathbb{R}^N$ is the input

* [Deep Inside Convolutional Networks, Simonyan, 2014](#)



SALIENCY



$F_C : R^N \rightarrow R^P$ - a NN Function

C - the prediction class

- Infinitesimal change in the neuron activations that changes the output

- $\Phi_c(F_L(x)) = \frac{\partial F_c(x)}{\partial F_L(x)}^*$, where $x \in \mathbb{R}^N$ is the input

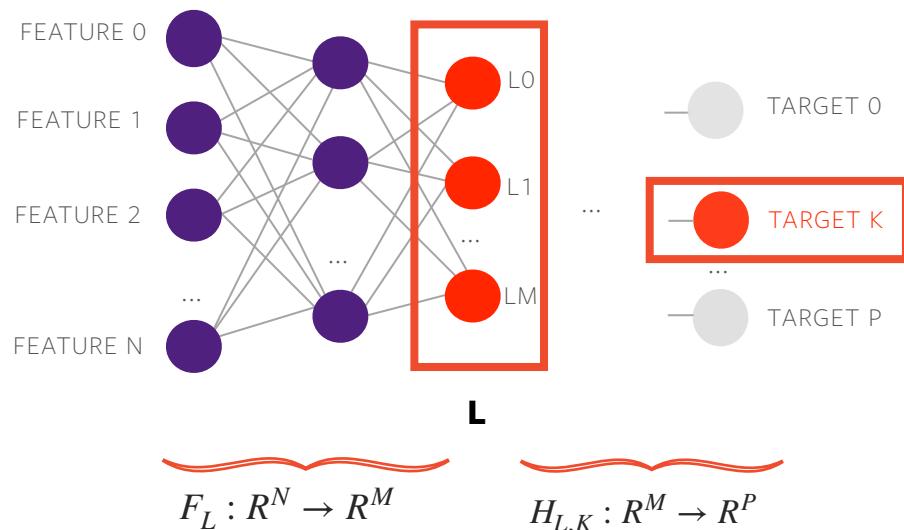
Limitations

- Gradients are noisy and sensitive to functions input

* [Deep Inside Convolutional Networks, Simonyan, 2014](#)



SMOOTHGRAD



$F_C : R^N \rightarrow R^P$ - a NN Function

C - the prediction class

- Adds noise to remove noise, [SmoothGrad, Smilkov 2017](#)
- Samples n examples in the neighborhood of input x and averages the explanations across all those examples

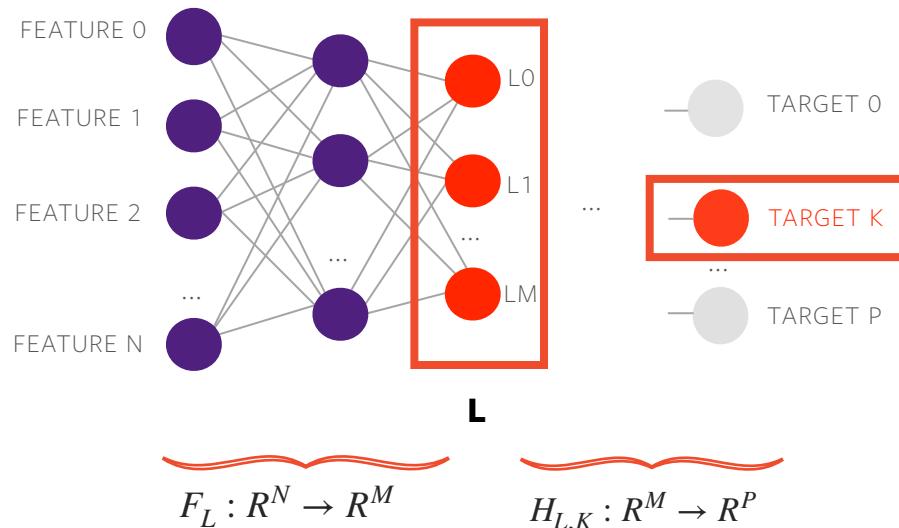
$$\hat{\Phi}_c(F_L(x)) = \frac{1}{n} \sum_0^n \Phi_c(F_L(x + \mathcal{N}(0, \sigma))),$$

σ is std

- Helps to stabilize explanations



SMOOTHGRAD



- Adds noise to remove noise, [SmoothGrad, Smilkov 2017](#)
 - Samples n examples in the neighborhood of input x and averages the explanations across all those examples
 - Helps to stabilize explanations
- $$\hat{\Phi}_c(F_L(x)) = \frac{1}{n} \sum_0^n \Phi_c(F_L(x + \mathcal{N}(0, \sigma))),$$
- σ is std

Limitations

- Finding optimal n can be challenging
- Computationally expensive depending on how large n is



INTEGRATED GRADIENTS

- Integrates the gradients along the path from baseline

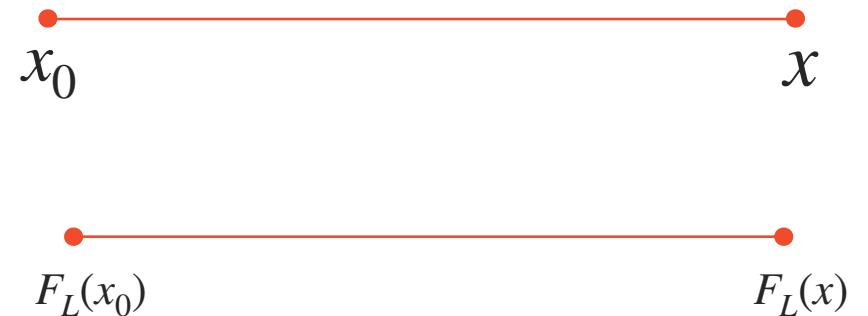
$x_0 \in \mathbb{R}^N$ to inputs $x \in \mathbb{R}^N$ ([Axiomatic Attribution for Deep Networks, Sundararajan, et. al., 2017](#))



INTEGRATED GRADIENTS

- Integrates the gradients along the path from baseline

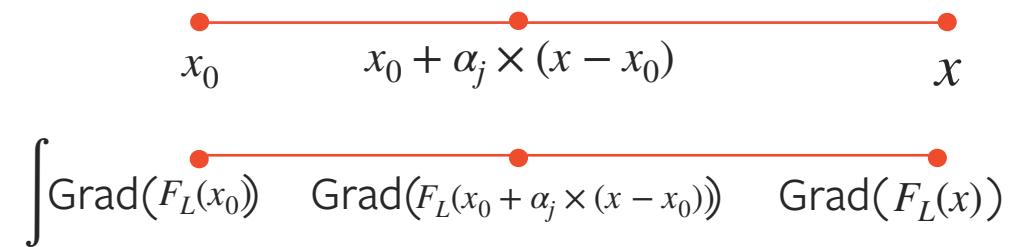
$x_0 \in \mathbb{R}^N$ to inputs $x \in \mathbb{R}^N$ ([Axiomatic Attribution for Deep Networks, Sundararajan, et. al., 2017](#))





INTEGRATED GRADIENTS

- Integrates the gradients along the path from baseline
 $x_0 \in \mathbb{R}^N$ to inputs $x \in \mathbb{R}^N$ ([Axiomatic Attribution for Deep Networks, Sundararajan, et. al., 2017](#))



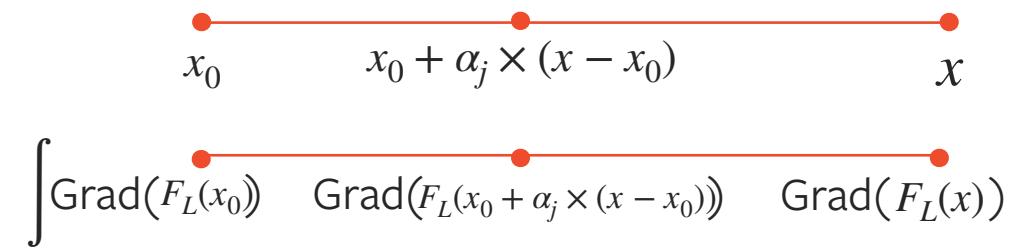


INTEGRATED GRADIENTS

- Integrates the gradients along the path from baseline

$x_0 \in \mathbb{R}^N$ to inputs $x \in \mathbb{R}^N$ ([Axiomatic Attribution for Deep Networks, Sundararajan, et. al., 2017](#))

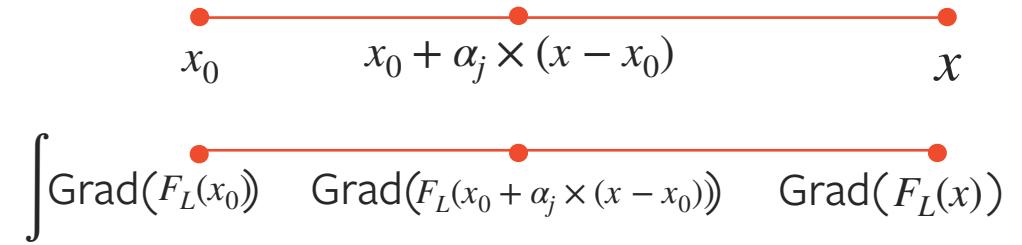
- $\Phi(F_L(x^i)) = (F_L(x^i) - F_L(x_0^i)) \cdot \int_0^1 \frac{\partial F(x_0^i + \alpha \cdot (x^i - x_0^i))}{\partial F_L(x^i)} d\alpha,$
where $\alpha \in [0,1]$ is a scaling factor





INTEGRATED GRADIENTS

- Integrates the gradients along the path from baseline $x_0 \in \mathbb{R}^N$ to inputs $x \in \mathbb{R}^N$ ([Axiomatic Attribution for Deep Networks, Sundararajan, et. al., 2017](#))
- $\Phi(F_L(x^i)) = (F_L(x^i) - F_L(x_0^i)) \cdot \int_0^1 \frac{\partial F(x_0^i + \alpha \cdot (x^i - x_0^i))}{\partial F_L(x^i)} d\alpha,$
where $\alpha \in [0,1]$ is a scaling factor



Limitations

- Finding good baselines can be challenging
- Feature correlations and interactions aren't taken into account



AXIOMS OF INTEGRATED GRADIENTS

- Completeness
 - Sum of the attributions is equal to the Neural Network (NN) function's differences at its input and baseline

$$\sum_{i=1}^n \Phi(x^i) = F(x) - F(x_0)$$



AXIOMS OF INTEGRATED GRADIENTS

- Completeness
 - Sum of the attributions is equal to the Neural Network (NN) function's differences at its input and baseline
$$\sum_{i=1}^n \Phi(x^i) = F(x) - F(x_0)$$
- Sensitivity
 - Sensitive to differing features and output predictions



AXIOMS OF INTEGRATED GRADIENTS

- Completeness
 - Sum of the attributions is equal to the Neural Network (NN) function's differences at its input and baseline
$$\sum_{i=1}^n \Phi(x^i) = F(x) - F(x_0)$$
- Sensitivity
 - Sensitive to differing features and output predictions
- Implementation Invariance
 - Attributions for two functionally equivalent NNs are identical



LAYER CONDUCTANCE

- Layer Conductance expands Integrated Gradients using chain rule in order to compute neuron importance ([How Important Is a Neuron?, Dhamdhere, et. al., 2018](#))



LAYER CONDUCTANCE

- Layer Conductance expands Integrated Gradients using chain rule in order to compute neuron importance ([How Important Is a Neuron?](#), Dhamdhere, et. al., 2018)
- Assume, $y = F_L(x^i)$, is a neuron in the output of layer L
- $\Phi(x_y^i) = (x^i - x_0^i) \cdot \int_0^1 \frac{\partial F(x_0^i + \alpha \cdot (x^i - x_0^i))}{\partial y} \frac{\partial y}{\partial x^i} d\alpha$
where $\alpha \in [0,1]$ is a scaling factor



LAYER CONDUCTANCE

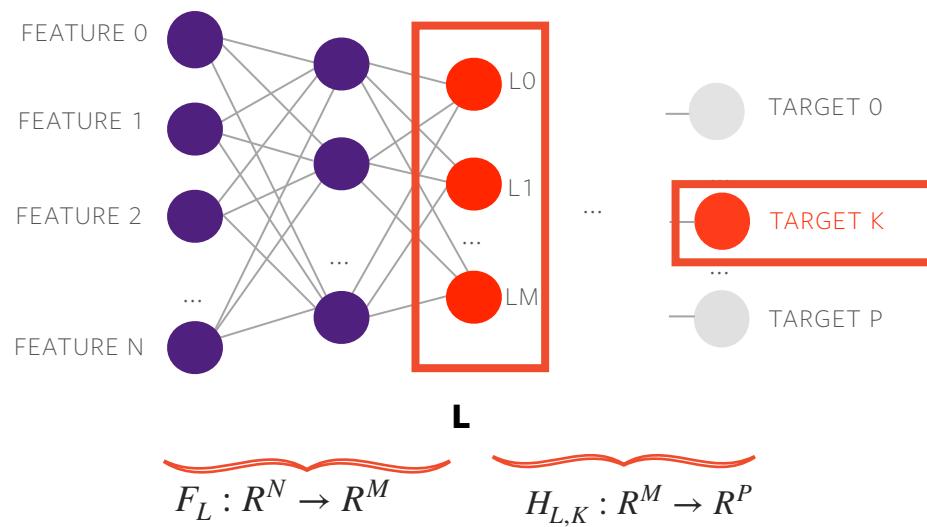
- Layer Conductance expands Integrated Gradients using chain rule in order to compute neuron importance ([How Important Is a Neuron?, Dhamdhere, et. al., 2018](#))
- Assume, $y = F_L(x^i)$, is a neuron in the output of layer L
- $\Phi(x_y^i) = (x^i - x_0^i) \cdot \int_0^1 \frac{\partial F(x_0^i + \alpha \cdot (x^i - x_0^i))}{\partial y} \frac{\partial y}{\partial x^i} d\alpha$

where $\alpha \in [0,1]$ is a scaling factor
- Axiomatic similar to Integrated gradients



NEURON ABLATION

- Measures the importance of neurons(s) based on the magnitude changes in prediction scores or measures of prediction goodness when neurons(s) are ablated





NEURON ABLATION

- Measures the importance of neurons(s) based on the magnitude changes in prediction scores or measures of prediction goodness when neurons(s) are ablated

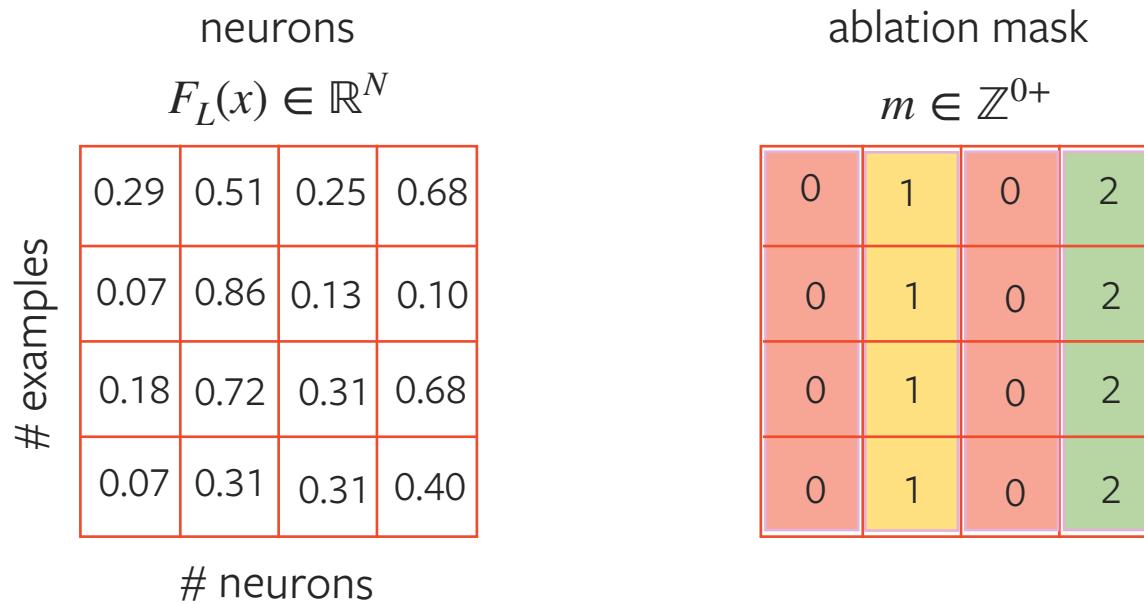
		neurons			
		$F_L(x) \in \mathbb{R}^N$			
		0.29	0.51	0.25	0.68
# examples		0.07	0.86	0.13	0.10
		0.18	0.72	0.31	0.68
		0.07	0.31	0.31	0.40

neurons



NEURON ABLATION

- Measures the importance of neurons(s) based on the magnitude changes in prediction scores or measures of prediction goodness when neurons(s) are ablated





NEURON ABLATION

- Measures the importance of neurons(s) based on the magnitude changes in prediction scores or measures of prediction goodness when neurons(s) are ablated

neurons				
# examples	$F_L(x) \in \mathbb{R}^N$			
	0.29	0.51	0.25	0.68
0.07	0.86	0.13	0.10	
0.18	0.72	0.31	0.68	
0.07	0.31	0.31	0.40	

ablation mask				
	$m \in \mathbb{Z}^{0+}$			
	0	1	0	2
0	1	0	2	
0	1	0	2	
0	1	0	2	

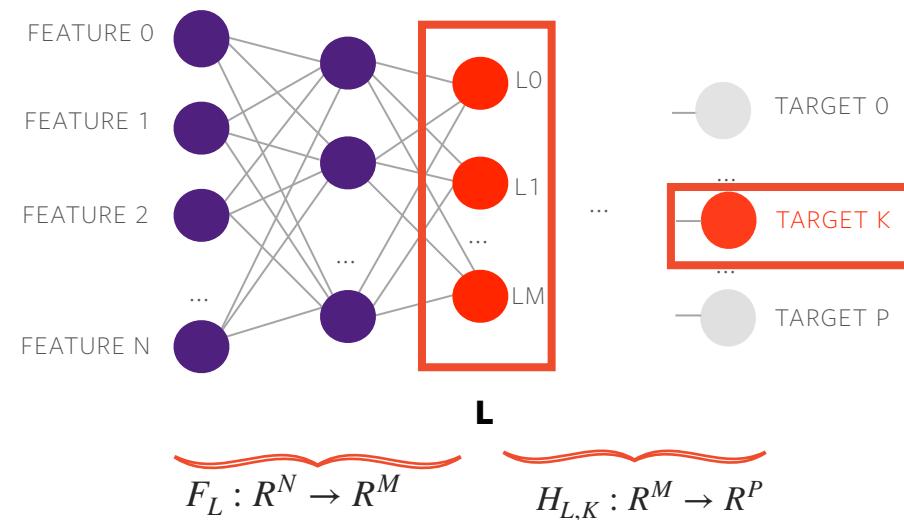
ablation baseline				
	$b \in \mathbb{R}^N$			
Default	0.0 0.0 0.0 0.0			
Custom	1.0 -1.0 1.0 0.0			



NEURON ABLATION

- Measures the importance of neurons(s) based on the magnitude changes in prediction scores or measures of prediction goodness when neurons(s) are ablated

$\Phi(F_L, H_{L,K}, x, m, b) = H_{L,K}(F_L(x)) - H_{L,K}(F_L(x), m, b)$, where $m \in \mathbb{Z}^{0+}$ is the ablation mask and $b \in \mathbb{R}^N$ ablation values





NEURON ABLATION

- Measures the importance of neurons(s) based on the magnitude changes in prediction scores or measures of prediction goodness when neurons(s) are ablated
- $\Phi(F_L, H_{L,K}, x, m, b) = H_{L,K}(F_L(x)) - H_{L,K}(F_L(x), m, b)$, where $m \in \mathbb{Z}^{0+}$ is the ablation mask and $b \in \mathbb{R}^N$ ablation values

Limitations

- Identifying which neurons mask together and what ablation values use
- Ablated neurons might lead to a model state which wasn't observed during train/valid phases of the network
- Computationally more expensive than gradient-based methods



FEATURE INTERACTION ATTRIBUTION



FEATURE INTERACTIONS

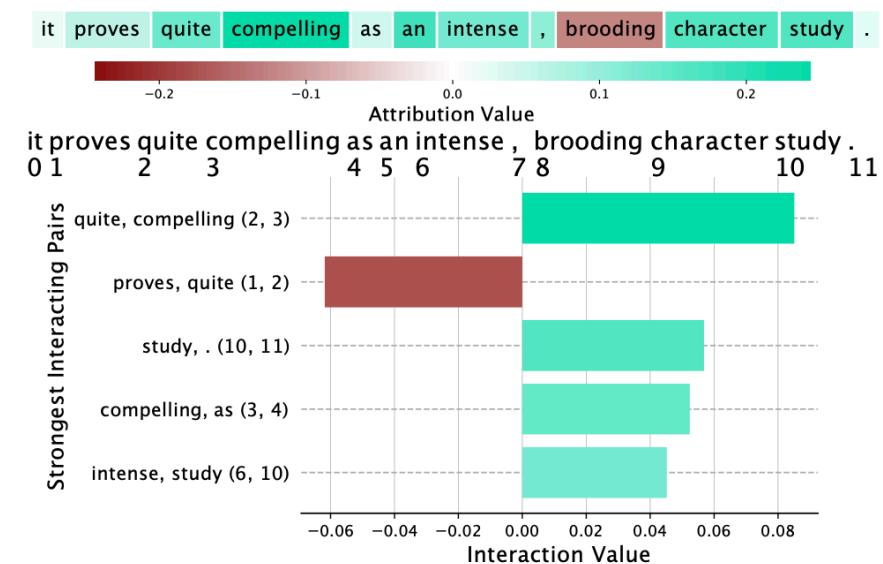
- The whole is different than the sum of its parts

$$\text{Attr_Score}('Not Interesting') \neq \text{Attr_Score}('Not') + \text{Attr_Score}('Interesting')$$



INTEGRATED HESSIANS

- An extension of integrated gradients that uses hessians to compute pairwise feature interaction importance
- An axiomatic method similar to Integrated Gradients
- Explains the importance of i-th feature on the importance of j-th feature



Source: Explaining Explanations: Axiomatic Feature Interactions for Deep Networks, Janizek, et.al., 2020



SHAPELY TAYLOR INDEX

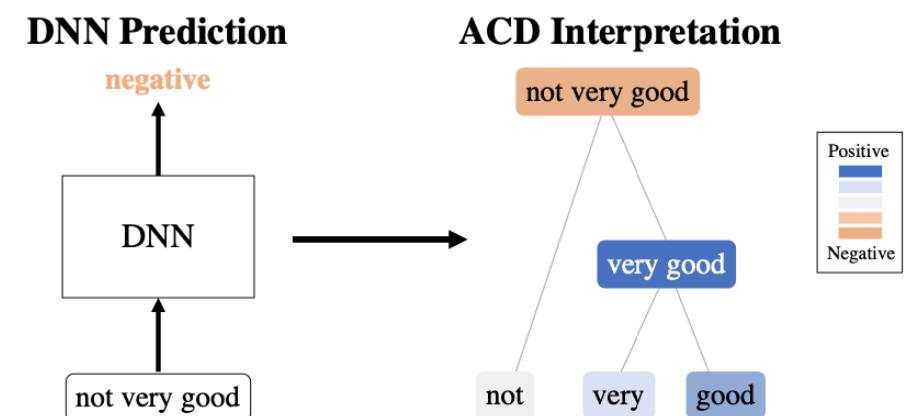
- A generalization of Shapely Values for feature interactions using higher order Taylor series terms
- Axiomatic similar to Shapely Values
- Computationally expensive depending on how exact the computations are

[Source: The Shapley Taylor Interaction Index, Dhamdhere, et.al., 2019](#)



AGGLOMERATIVE CONTEXTUAL DECOMPOSITION (ACD)

- Explains model predictions using hierarchical clusters of input features
- It is based on the additive decomposition of DNN with adjustments for different types of activations such as MaxPool or Convolution
- It's not axiomatic as previous two methods

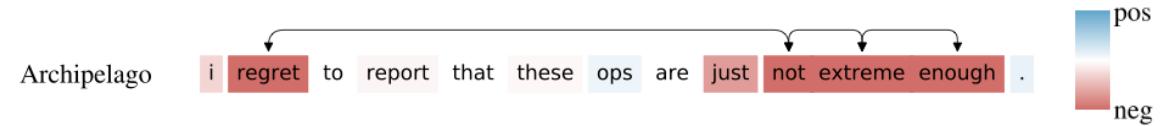


Source: Hierarchical Interpretations for Neural Network Predictions, Singh, et.al., 2019



ARCHIPELAGO

- Two step procedure
 - **ArchDetect:** Identifies k-th order feature interactions
 - **ArchInterpret:** Estimates feature importance of the interacted feature sets
 - Axiomatic similar to Integrated Hessians and Shapely Taylor Index





EVALUATION OF ATTRIBUTION METHODS



EVALUATION OF ATTRIBUTION METHODS

- No clear guidance on how to quantify the quality of attributions
- Quantitative metrics are often domain specific
- Visual evaluation can be misleading or seen as a confirmation bias



SENSITIVITY - MAX

- Measures the sensitivity of attributions to subtle input perturbations using Monte-Carlo sampling-based approximation ([On the \(In\)fidelity and Sensitivity of Explanations, Yeh, et. al., 2019](#))
- Given input $x \in \mathbb{R}^N$, perturbed input $y \in \mathbb{R}^N$, perturbation radius $r \in \mathbb{R}$, a NN function $F : \mathbb{R}^N \rightarrow \mathbb{R}$ and an explanation function $\Phi : F \times \mathbb{R}^N \rightarrow \mathbb{R}^N$

$$SENS_{MAX}(\Phi, F, x, r) = \max_{||y-x|| \leq r} \frac{||\Phi(F, y) - \Phi(F, x)||}{||\Phi(F, x)||}$$



INFIDELITY

- Measures mean-squared error between dot product of input perturbation and attribution and differences between the predictor function at its input and perturbed input ([On the \(In\)fidelity and Sensitivity of Explanations, Yeh, et. al., 2019](#))
- Completeness property is a special case of infidelity metric

Given input $x \in \mathbb{R}^N$, a NN function $F : \mathbb{R}^N \rightarrow \mathbb{R}$, a meaningful perturbation $I \in \mathbb{R}^N$ with a probability measure μ_I

$$INFD_{\mu_I}(\Phi, F, x) = \mathbb{E}_{I \sim \mu_I}[(I^T \Phi(F, x) - (F(x) - F(x - I)))^2]$$



LIBRARIES FOR EXPLAINABILITY, FEATURE AND NEURON ATTRIBUTION

InterpretML - Alpha Release

license MIT | python 3.6 | 3.7 | 3.8 | pypi v0.2.4 | build failing | coverage 95% | code quality: python A | maintained yes

In the beginning machines learned in darkness, and data scientists struggled in the void to explain them.

Let there be light.

AI Explainability 360

This extensible open source toolkit can help you comprehend how machine learning models predict labels by various means throughout the AI application lifecycle. We invite you to use it and improve it.

Captum
Model Interpretability for PyTorch

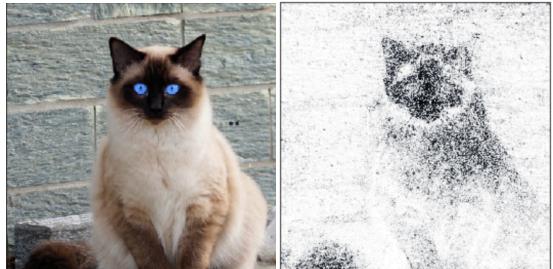
INTRODUCTION GET STARTED TUTORIALS





A MODEL INTERPRETABILITY LIBRARY FOR PYTORCH

MULTIMODAL



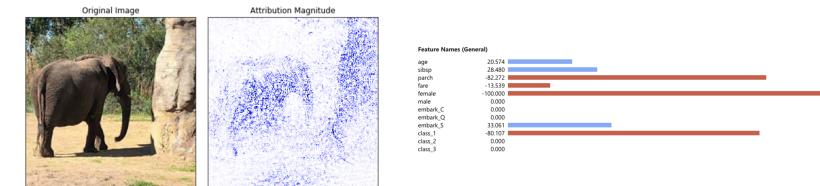
What color are the cats eyes?
Predicted
Blue (0.517)

EXTENSIBLE

```
class MyAttribution(Attribution):  
  
    def attribute(self, input, ...):  
        attributions = self._compute_attrs(input, ... )  
        # <Add any logic necessary for attribution>  
        return attributions
```

EASY TO USE

```
visualize_image_attr(attr.algo.attribute(input), ...)
```



this movie is awful . just awful . someone bought it for me as a christmas present because they knew i liked a good horror flick . i do n't think they understood the "good" part . all i can say is next year this person is getting slipper socks from me . avoid this movie - it makes you bitter . peace.



WHAT DOES THE CAPTUM LIBRARY OFFER ?

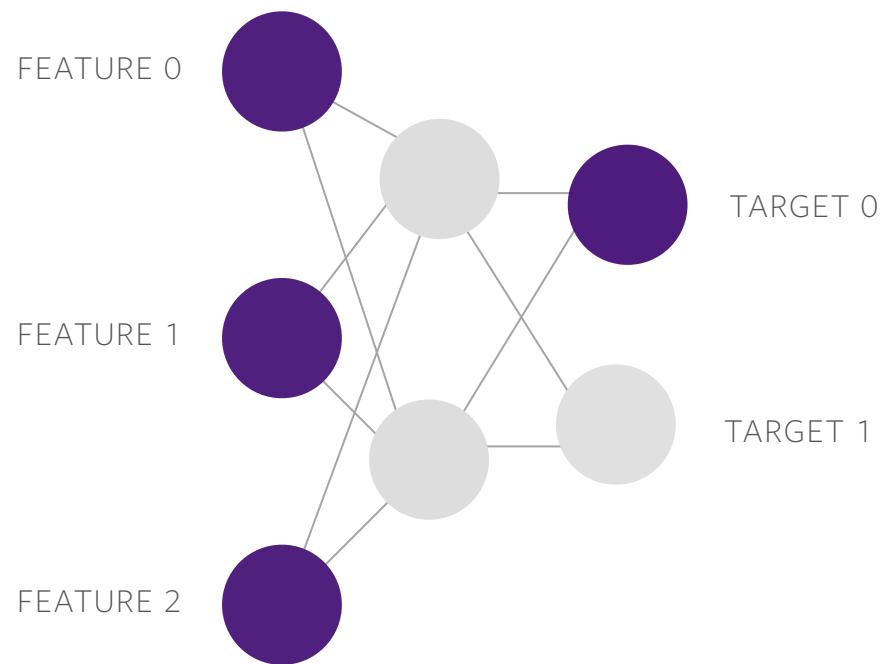
A number of gradient and perturbation-based attribution algorithms to interpret:



WHAT DOES THE CAPTUM LIBRARY OFFER ?

A number of gradient and perturbation-based attribution algorithms
to interpret:

- **Output predictions with respect to inputs**
- Output predictions with respect to all neurons in the layers
- Neurons with respect to inputs

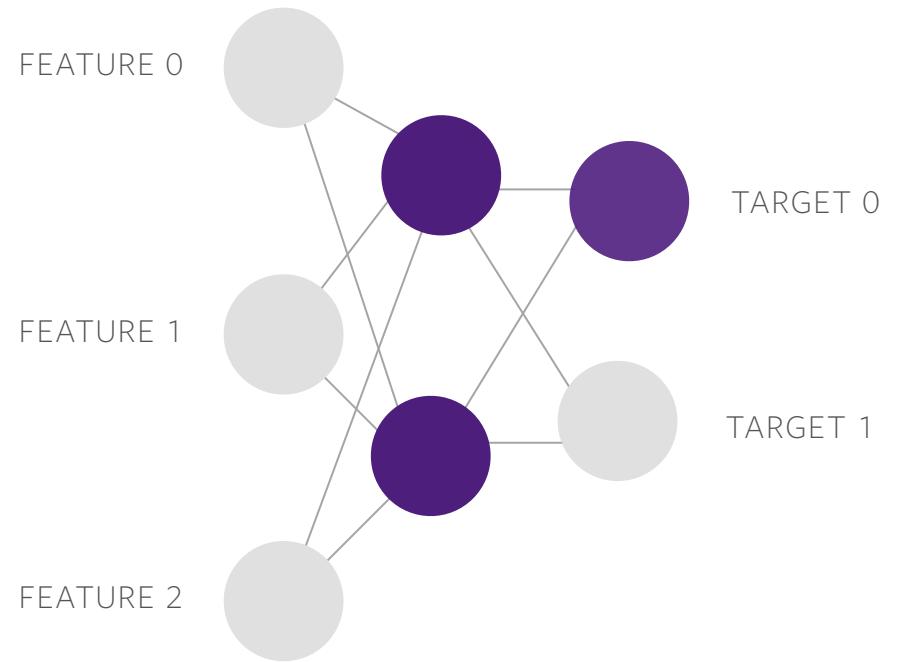




WHAT DOES THE CAPTUM LIBRARY OFFER ?

A number of gradient and perturbation-based attribution algorithms
to interpret:

- Output predictions with respect to inputs
- **Output predictions with respect to all neurons in the layers**
- Neurons with respect to inputs

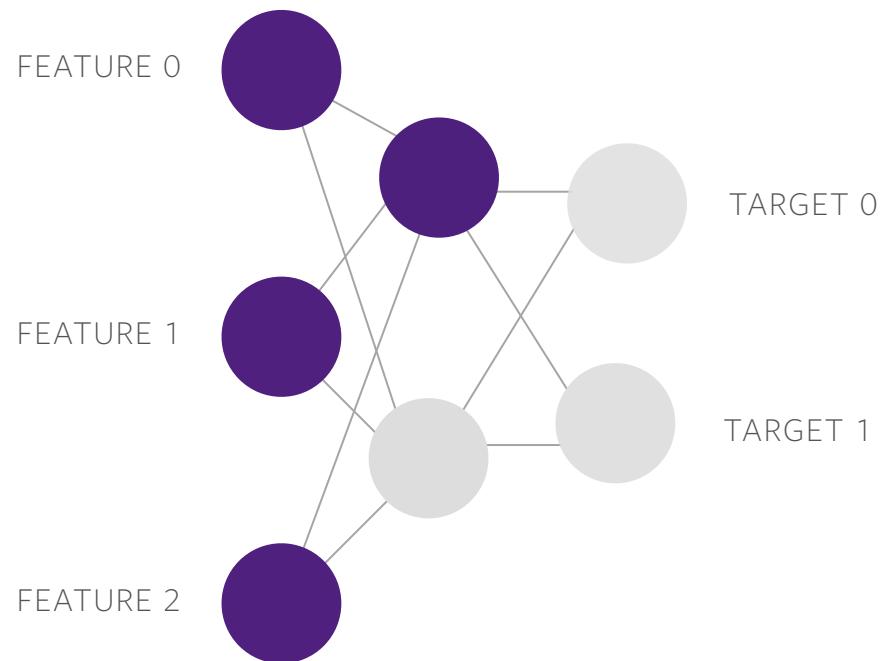




WHAT DOES THE CAPTUM LIBRARY OFFER ?

A number of gradient and perturbation-based attribution algorithms
to interpret:

- Output predictions with respect to inputs
- Output predictions with respect to all neurons in the layers
- **Neurons with respect to inputs**





WHAT DOES THE CAPTUM LIBRARY OFFER ?

A number of gradient and perturbation-based attribution algorithms
to interpret:

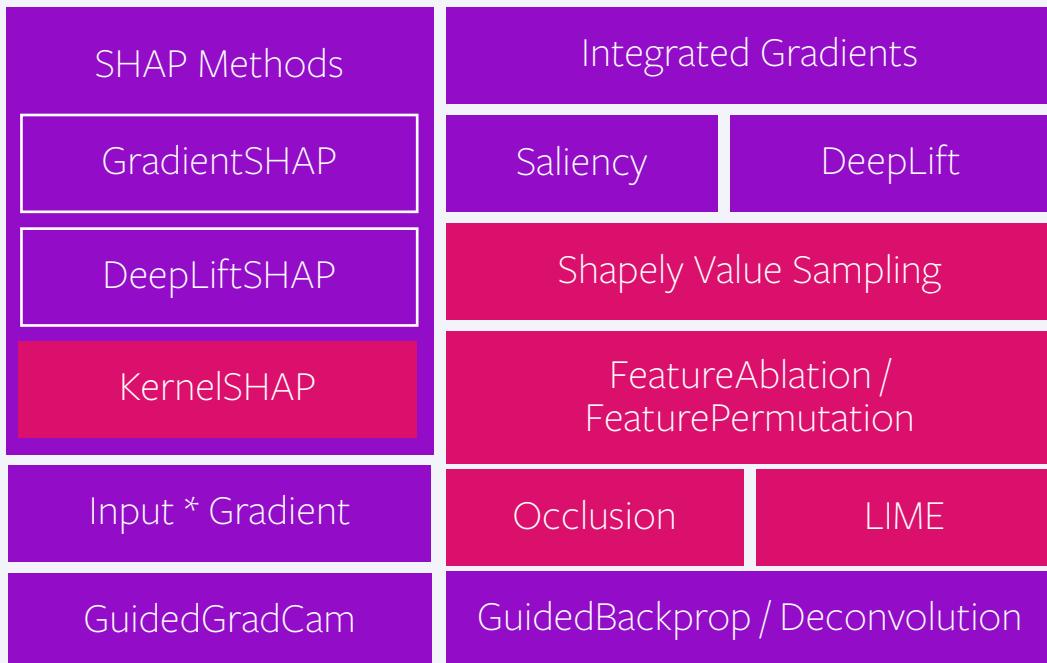
- Output predictions with respect to inputs
 - Output predictions with respect to all neurons in the layers
 - Neurons with respect to inputs
-
- **Infidelity and Sensitivity-Max evaluation metrics**



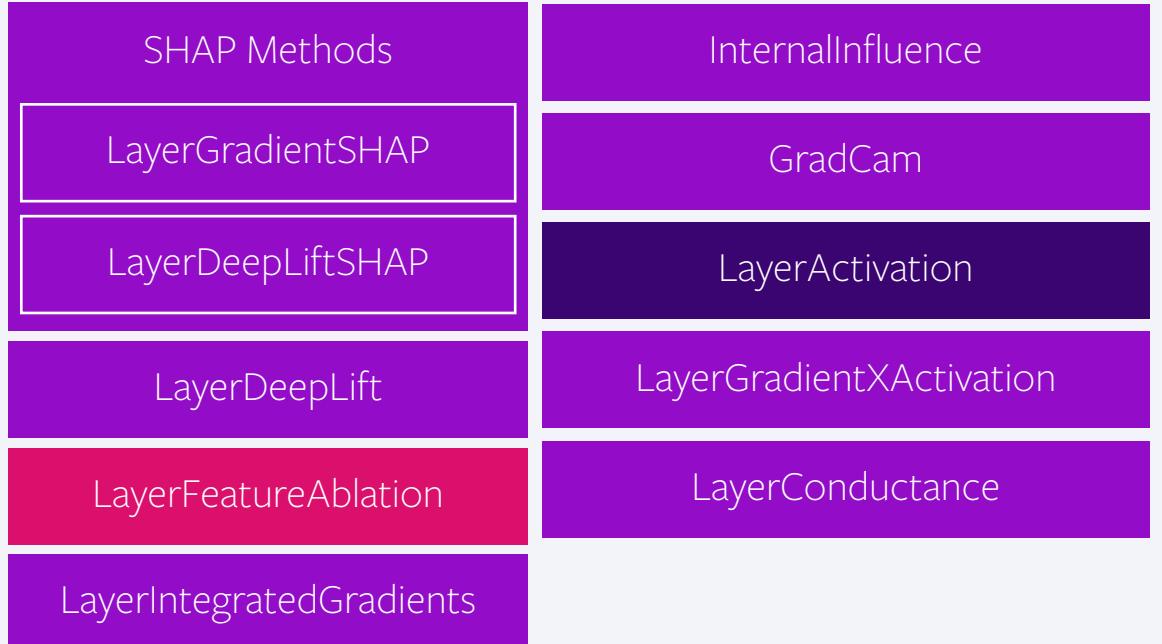
ATTRIBUTION ALGORITHMS



Attribute model output (or internal neurons) to input features



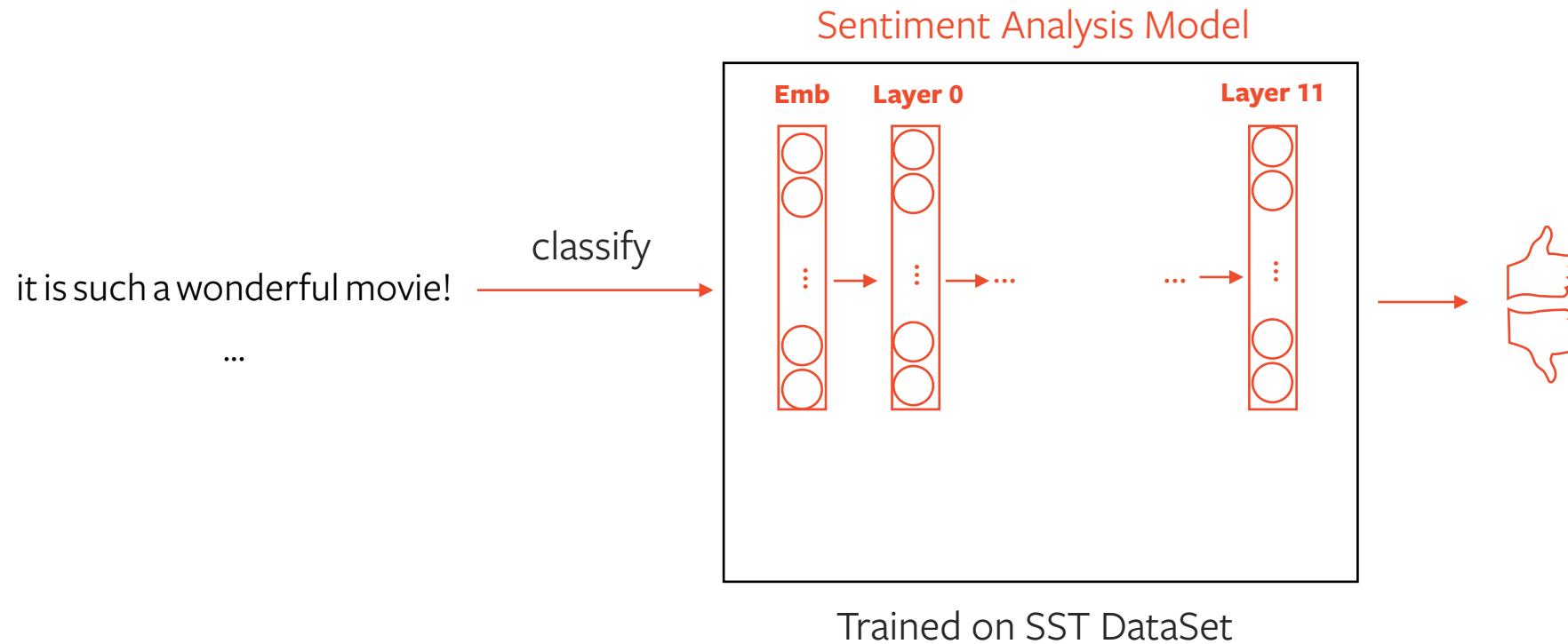
Attribute model output to the layers of the model



NoiseTunnel (Smoothgrad, Vargrad, Smoothgrad Square)

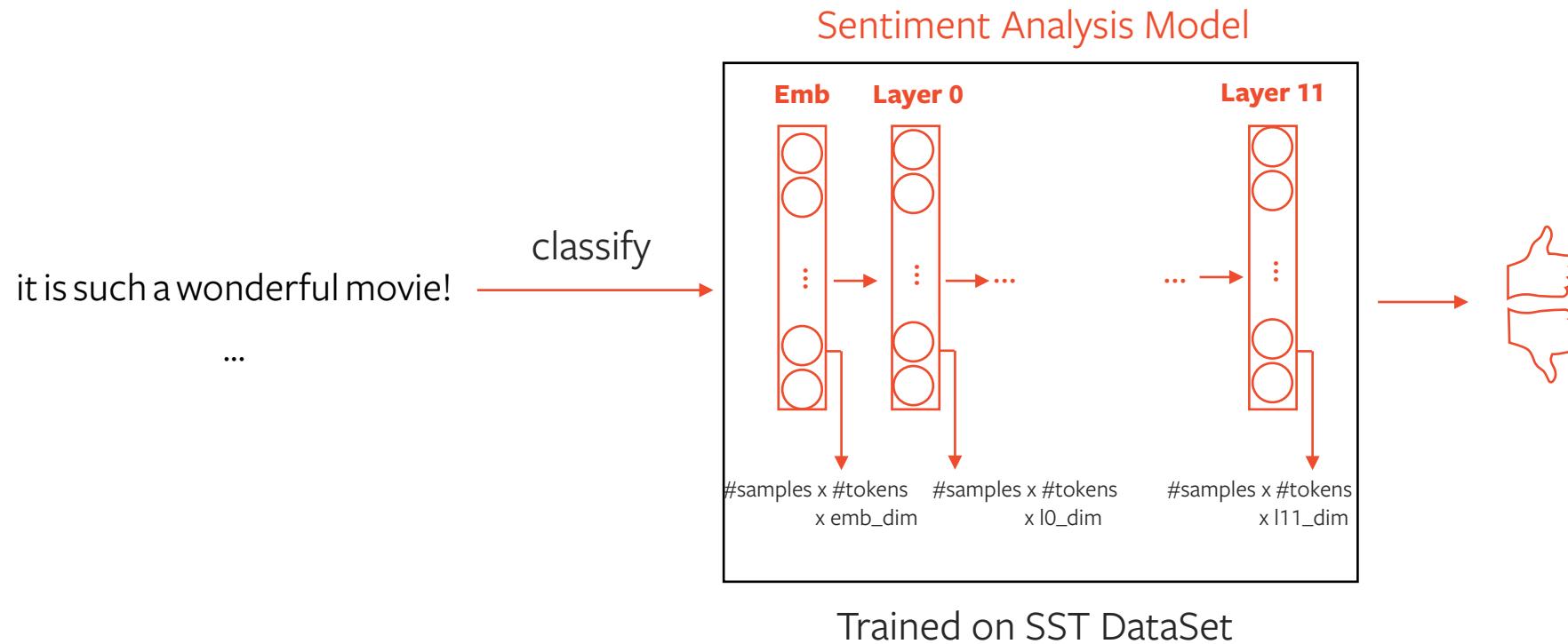


INTERPRETING THE PREDICTIONS OF SENTIMENT CLASSIFICATION USING BERT





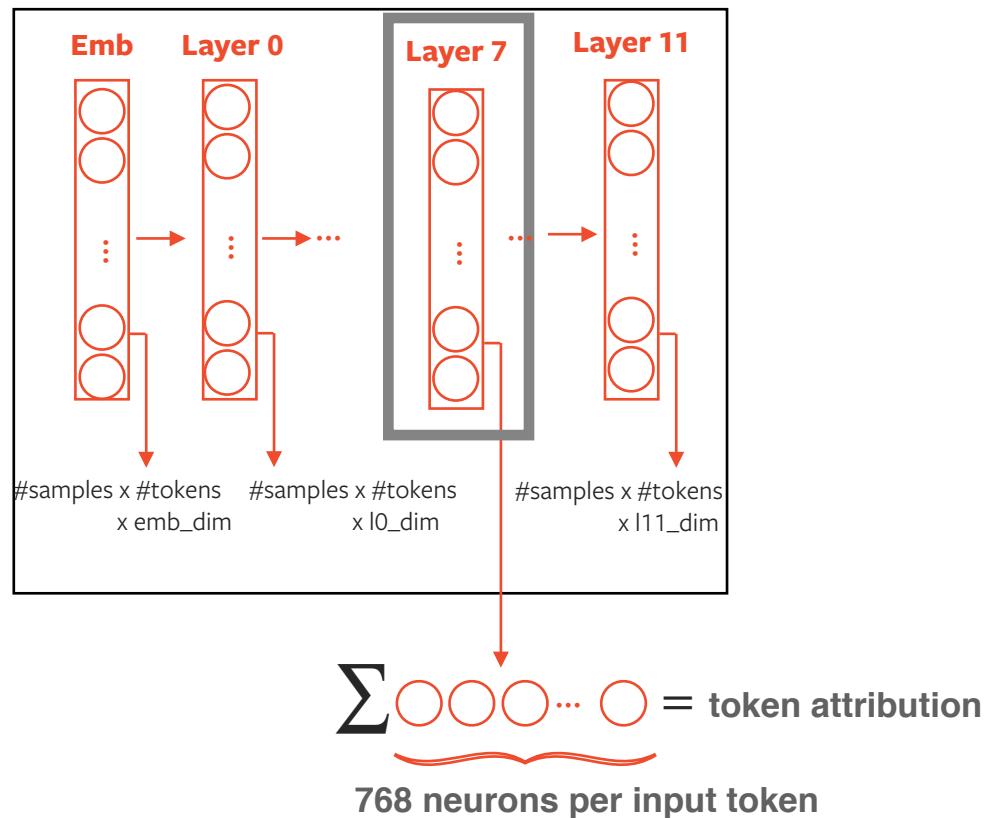
INTERPRETING THE PREDICTIONS OF SENTIMENT CLASSIFICATION USING BERT





LAYERGRADIENTS

Sentiment Analysis Model



```
from captum.attr import LayerGradientXActivation
from captum.attr import visualization as viz

attr_algo = LayerGradientXActivation(model,
                                       model.bert.encoder.layer[7],
                                       multiply_by_inputs=False)

# Computing the attributions for plane w.r.t. inputs
attrs = attr_algo.attribute(sentence_tensor,
                            target = 1)

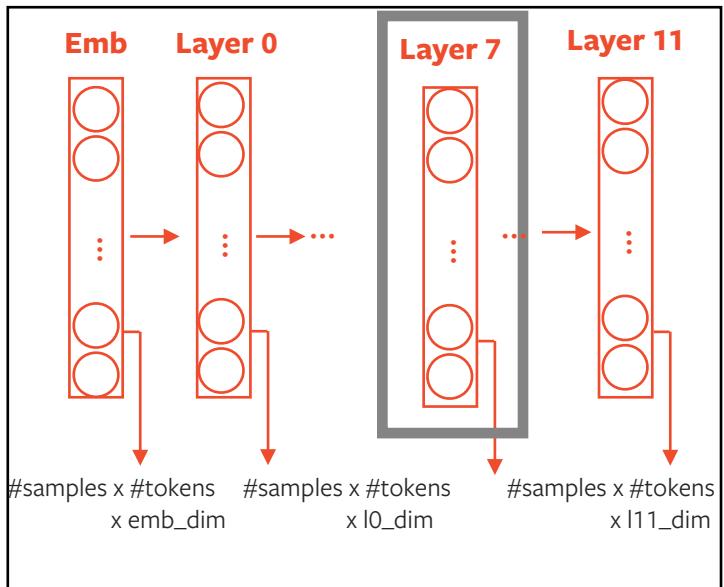
# Sum attributions along embedding dimension
attrs_tokens = attrs.sum(axis=-1).squeeze(0)

# Visualizing attributions
viz.visualize_text([viz.VisualizationDataRecord(
    attrs_tokens,
    torch.max(preds),
    ...)])
```



LAYERGRADIENTS

Sentiment Analysis Model



Attributions

[CLS] it is such a wonderful movie [SEP] [PAD] [PAD]

Legend: ■ Negative □ Neutral ■ Positive

```
from captum.attr import LayerGradientXActivation
from captum.attr import visualization as viz

attr_algo = LayerGradientXActivation(model,
                                       model.bert.encoder.layer[7],
                                       multiply_by_inputs=False)

# Computing the attributions for plane w.r.t. inputs
attrs = attr_algo.attribute(sentence_tensor,
                            target = 1)

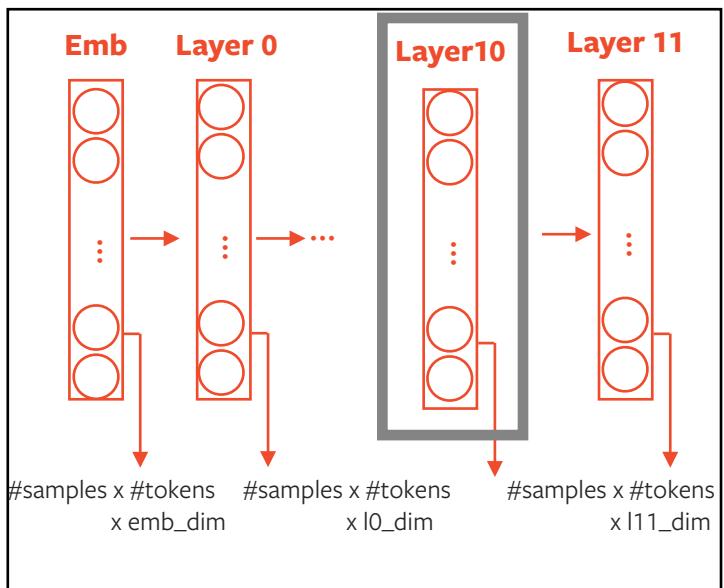
# Sum attributions along embedding dimension
attrs_tokens = attrs.sum(axis=-1).squeeze(0)

# Visualizing attributions
viz.visualize_text([viz.VisualizationDataRecord(
                     attrs_tokens,
                     torch.max(preds),
                     ...)])
```



LAYERGRADIENTS

Sentiment Analysis Model



Attributions

[CLS] it is such a wonderful movie [SEP] [PAD] [PAD]

Legend: ■ Negative □ Neutral ■ Positive

```
from captum.attr import LayerGradientXActivation
from captum.attr import visualization as viz

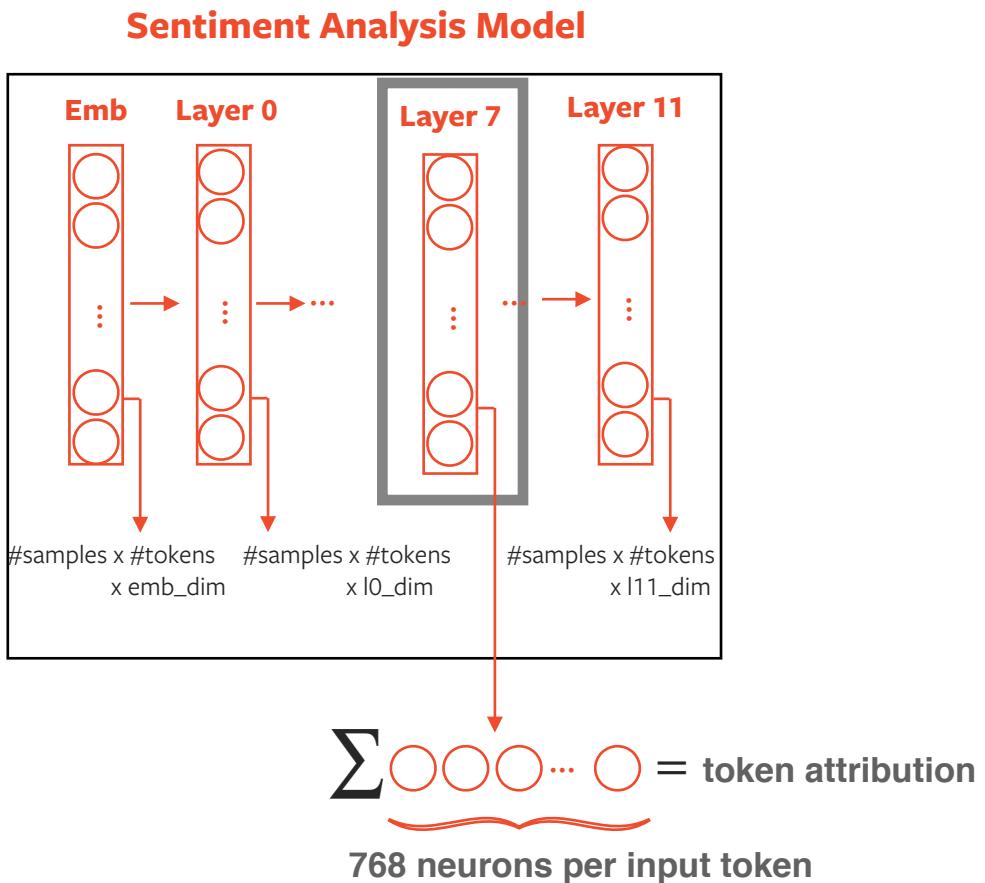
attr_algo = LayerGradientXActivation(model,
                                       model.bert.encoder.layer[10],
                                       multiply_by_inputs=False)

# Computing the attributions for plane w.r.t. inputs
attrs = attr_algo.attribute(sentence_tensor,
                            target = 1)

# Sum attributions along embedding dimension
attrs_tokens = attrs.sum(axis=-1).squeeze(0)

# Visualizing attributions
viz.visualize_text([viz.VisualizationDataRecord(
                     attrs_tokens,
                     torch.max(preds),
                     ...)])
```

LAYER INTEGRATED GRADIENTS



```
from captum.attr import LayerIntegratedGradients
from captum.attr import visualization as viz

attr_algo = LayerIntegratedGradients(model,
                                       model.bert.encoder.layer[7])

# Computing the attributions for plane w.r.t. inputs
attrs = attr_algo.attribute(sentence_tensor,
                            target = 1)

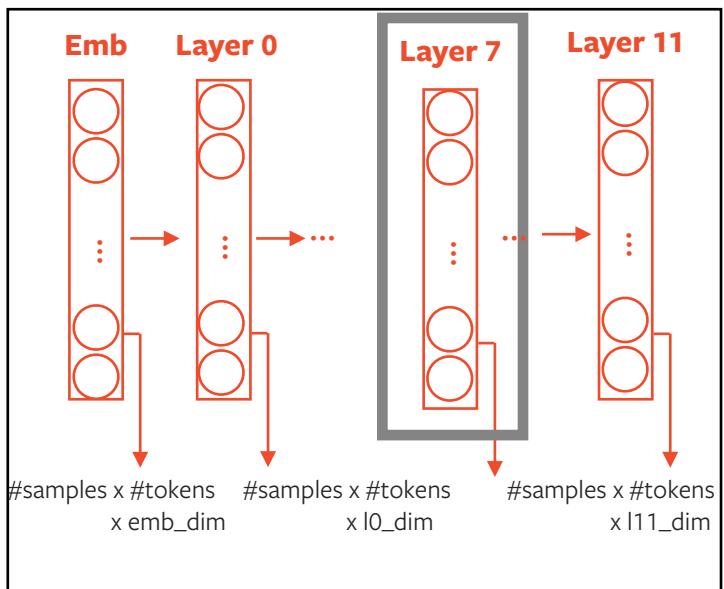
# Sum attributions along embedding dimension
attrs_tokens = attrs.sum(axis=-1).squeeze(0)

# Visualizing attributions
viz.visualize_text([viz.VisualizationDataRecord(
    attrs_tokens,
    torch.max(preds),
    ...)])
```



LAYER INTEGRATED GRADIENTS

Sentiment Analysis Model



Attributions

CLS] it is such a wonderful movie [SEP] [PAD] [PAD]

Legend: ■ Negative □ Neutral ■ Positive

```
from captum.attr import LayerIntegratedGradients
from captum.attr import visualization as viz

attr_algo = LayerIntegratedGradients(model,
                                       model.bert.encoder.layer[7])

# Computing the attributions for plane w.r.t. inputs
attrs = attr_algo.attribute(sentence_tensor,
                            target = 1)

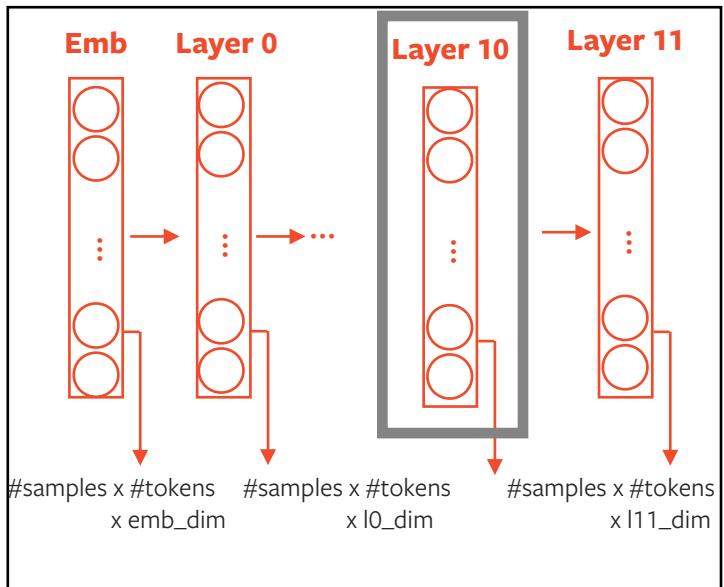
# Sum attributions along embedding dimension
attrs_tokens = attrs.sum(axis=-1).squeeze(0)

# Visualizing attributions
viz.visualize_text([viz.VisualizationDataRecord(
                     attrs_tokens,
                     torch.max(preds),
                     ...)])
```



LAYER INTEGRATED GRADIENTS

Sentiment Analysis Model



Attributions

[CLS] it is such a wonderful movie [SEP] [PAD] [PAD]

Legend: ■ Negative □ Neutral ■ Positive

```
from captum.attr import LayerIntegratedGradients
from captum.attr import visualization as viz

attr_algo = LayerIntegratedGradients(model,
                                       model.bert.encoder.layer[10])

# Computing the attributions for plane w.r.t. inputs
attrs = attr_algo.attribute(sentence_tensor,
                            target = 1)

# Sum attributions along embedding dimension
attrs_tokens = attrs.sum(axis=-1).squeeze(0)

# Visualizing attributions
viz.visualize_text([viz.VisualizationDataRecord(
                     attrs_tokens,
                     torch.max(preds),
                     ...)])
```

```
attributions = LayerAttribution(forward_func, layers,...)  
    .attribute(input, ...)
```



A CASE STUDY FOR BERT QUESTION-ANSWERING MODEL



EXPLAINING BERT MODELS

- + Finetuning BERT model for Question Answering on SQuAD dataset
- + Evaluating on Dev Set

Exact Match: 78%

F1-Score: 86%





EXPLAINING BERT MODELS

- + Finetuning BERT model for Question Answering on SQuAD dataset
- + Evaluating on Dev Set
 - Exact Match: 78%
 - F1-Score: 86%
- + Understanding the importance of different types of word tokens, layers and neurons





EXISTING WORK ON BERT MODEL UNDERSTANDING

What does BERT look at? An Analysis of BERT's Attention

Kevin Clark[†] Urvashi Khandelwal[†] Omer Levy[‡] Christopher D. Manning[†]

[†]Computer Science Department, Stanford University

[‡]Facebook AI Research

{kevclark, urvashik, manning}@cs.stanford.edu
omerlevy@fb.com

Revealing the Dark Secrets of BERT

Olga Kovaleva, Alexey Romanov, Anna Rogers, Anna Rumshisky

Department of Computer Science
University of Massachusetts Lowell
Lowell, MA 01854

{okovalev, arum, aromanov}

Attention is Not Only a Weight: Analyzing Transformers with Vector Norms

Goro Kobayashi¹ Tatsuki Kurabayashi^{1,2} Sho Yokoi^{1,3} Kentaro Inui^{1,3}

¹ Tohoku University ² Langsmith Inc. ³ RIKEN

{goro.koba, kurabayashi, yokoi, inui}@ecei.tohoku.ac.jp

EXBERT: A Visual Analysis Tool to Explore Learned Representations in Transformers Models

Ben Hoover

IBM Research
MIT-IBM Lab

benjamin.hoover@ibm.com

Hendrik Strobelt

IBM Research
MIT-IBM Lab

hendrik.strobelt@ibm.com

Sebastian Gehrmann

Harvard SEAS

gehrmann@seas.harvard.edu

Self-Attention Attribution: Interpreting Information Interactions Inside Transformer

Yaru Hao,^{1,2*} Li Dong,² Furu Wei,² Ke Xu¹

¹ Beihang University

² Microsoft Research

haoyaru@,kexu@nlsde.{}buaa.edu.cn

{lidong1,fuwei}@microsoft.com



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
text = 'It is important to us to include, empower and support humans of all kinds.'
```



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
text = 'It is important to us to include, empower and support humans of all kinds.'  
question = 'What is important to us?'
```



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
text = 'It is important to us to include, empower and support humans of all kinds.'  
question = 'What is important to us?'
```

tokens

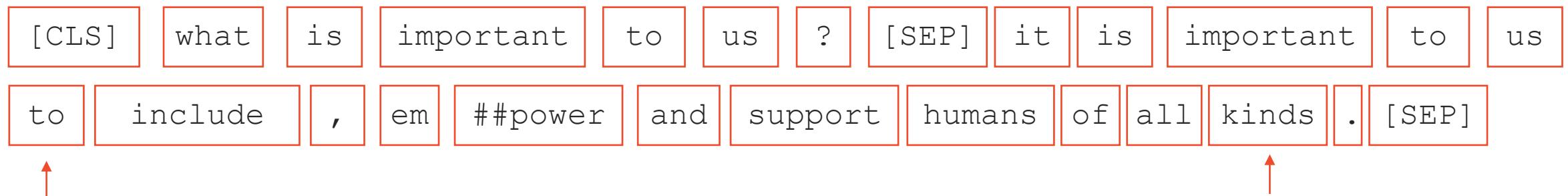
[CLS]	what	is	important	to	us	?	[SEP]	it	is	important	to	us
to	include	,	em	#power	and	support	humans	of	all	kinds	.	[SEP]



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
text = 'It is important to us to include, empower and support humans of all kinds.'  
question = 'What is important to us?'
```

tokens



P(Start Position) = 0.52

P(End Position) = 0.75



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
text = 'It is important to us to include, empower  
and support humans of all kinds.'  
  
question = 'What is important to us?'  
  
# load pre-trained model  
model = BertForQuestionAnswering.from_pretrained(path)
```

```
BertForQuestionAnswering(  
    bert): BertModel(  
        embeddings): BertEmbeddings(  
            word_embeddings): Embedding(28996, 768, padding_idx=0)  
            position_embeddings): Embedding(512, 768)  
            token_type_embeddings): Embedding(2, 768)  
            LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
            dropout): Dropout(p=0.1, inplace=False)  
        )  
        encoder): BertEncoder(  
            layer): ModuleList(  
                0): BertLayer(  
                    attention): BertAttention(  
                        self): BertSelfAttention(  
                            query): Linear(in_features=768, out_features=768, bias=True)  
                            key): Linear(in_features=768, out_features=768, bias=True)  
                            value): Linear(in_features=768, out_features=768, bias=True)  
                            dropout): Dropout(p=0.1, inplace=False)  
                        )  
                    )  
                    intermediate): BertIntermediate(  
                        dense): Linear(in_features=768, out_features=3072, bias=True)  
                    )  
                    output): BertOutput(  
                        dense): Linear(in_features=3072, out_features=768, bias=True)  
                        LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
                        dropout): Dropout(p=0.1, inplace=False)  
                    )  
                )  
                1): BertLayer(  
                    attention): BertAttention(  
                        self): BertSelfAttention(  
                            query): Linear(in_features=768, out_features=768, bias=True)  
                            key): Linear(in_features=768, out_features=768, bias=True)  
                            value): Linear(in_features=768, out_features=768, bias=True)  
                            dropout): Dropout(p=0.1, inplace=False)  
                        )  
                    )  
                )  
            )  
        )  
    )  
)
```



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
# A custom forward function to return either the start
# (pos = 0) or the end (pos = 1) prediction of the
# answer
def squad_pos_forward_func(inputs, token_type, ...,
                            position=0):
    predictions = model(inputs, ... )[position]

    return predictions.max(1).values

lig = LayerIntegratedGradients(squad_pos_forward_func,
                                layer=model.bert.embeddings)
```

```
BertForQuestionAnswering(
    bert): BertModel(
        embeddings): BertEmbeddings(
            word_embeddings): Embedding(28996, 768, padding_idx=0)
            position_embeddings): Embedding(512, 768)
            token_type_embeddings): Embedding(2, 768)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
        )
        (encoder): BertEncoder(
            (layer): ModuleList(
                (0): BertLayer(
                    (attention): BertAttention(
                        (self): BertSelfAttention(
                            (query): Linear(in_features=768, out_features=768, bias=True)
                            (key): Linear(in_features=768, out_features=768, bias=True)
                            (value): Linear(in_features=768, out_features=768, bias=True)
                            (dropout): Dropout(p=0.1, inplace=False)
                        )
                    )
                    (intermediate): BertIntermediate(
                        (dense): Linear(in_features=768, out_features=3072, bias=True)
                    )
                    (output): BertOutput(
                        (dense): Linear(in_features=3072, out_features=768, bias=True)
                        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
                        (dropout): Dropout(p=0.1, inplace=False)
                    )
                )
                (1): BertLayer(
                    (attention): BertAttention(
                        (self): BertSelfAttention(
                            (query): Linear(in_features=768, out_features=768, bias=True)
                            (key): Linear(in_features=768, out_features=768, bias=True)
                            (value): Linear(in_features=768, out_features=768, bias=True)
                            (dropout): Dropout(p=0.1, inplace=False)
                        )
                    )
                )
            )
        )
    )
)
```



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
# A custom forward function to return either the start
# (pos = 0) or the end (pos = 1) prediction of the
# answer
def squad_pos_forward_func(inputs, token_type, ...,
                            pos=0):
    prediction = model(inputs, ...)[pos]

    return prediction.max(1).values

lig = LayerIntegratedGradients(squad_pos_forward_func,
                                layer=model.bert.embeddings)

# attributing to start position
attr_start = lig.attribute(inputs=input_ids,
                           baselines=ref_input_ids,
                           additional_forward_args=(mask, 0))
```

```
BertForQuestionAnswering(
  bert): BertModel(
    embeddings): BertEmbeddings(
      word_embeddings): Embedding(28996, 768, pa
      position_embeddings): Embedding(512, 768)
      token_type_embeddings): Embedding(2, 768)
      LayerNorm): LayerNorm((768,), eps=1e-12, e
      dropout): Dropout(p=0.1, inplace=False)
    )
    encoder): BertEncoder(
      layer): ModuleList(
        0): BertLayer(
          attention): BertAttention(
            query): BertAttention(
              query_weight): BertWeight(
                query_weight): BertWeight(
                  query_weight): BertWeight(
                    query_weight): BertWeight(
                      query_weight): BertWeight(
                        query_weight): BertWeight(
                          query_weight): BertWeight(
                            query_weight): BertWeight(
                              query_weight): BertWeight(
                                query_weight): BertWeight(
                                  query_weight): BertWeight(
                                    query_weight): BertWeight(
                                      query_weight): BertWeight(
                                        query_weight): BertWeight(
                                          query_weight): BertWeight(
                                            query_weight): BertWeight(
                                              query_weight): BertWeight(
                                                query_weight): BertWeight(
                                                  query_weight): BertWeight(
                                                    query_weight): BertWeight(
                                                      query_weight): BertWeight(
                                                        query_weight): BertWeight(
                                                          query_weight): BertWeight(
                                                            query_weight): BertWeight(
                                                              query_weight): BertWeight(
                                                                query_weight): BertWeight(
                                                                  query_weight): BertWeight(
                                                                    query_weight): BertWeight(
                                                                      query_weight): BertWeight(
                                                                        query_weight): BertWeight(
                                                                          query_weight): BertWeight(
                                                                            query_weight): BertWeight(
                                                                              query_weight): BertWeight(
                                                                                query_weight): BertWeight(
                                                                                  query_weight): BertWeight(
                                                                                    query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
................................................................
```

[CLS] what is important to us ? [SEP] it is important to us to include , em ##power and support humans of all kinds . [SEP]

Highly attributed for start position

BertEmbedding

important
?
what
is
us



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
# A custom forward function to return either the start
# (pos = 0) or the end (pos = 1) prediction of the
# answer
def squad_pos_forward_func(inputs, token_type, ...,
                            pos=0):
    prediction = model(inputs, ...)[pos]

    return prediction.max(1).values

lig = LayerIntegratedGradients(squad_pos_forward_func,
                                layer=model.bert.embeddings)

# attributing to end position
attr_end = lig.attribute(inputs=input_ids,
                        baselines=ref_input_ids,
                        additional_forward_args=(mask, 1))
```

```
BertForQuestionAnswering(
  bert): BertModel(
    embeddings): BertEmbeddings(
      word_embeddings): Embedding(28996, 768, pa
      position_embeddings): Embedding(512, 768)
      token_type_embeddings): Embedding(2, 768)
      LayerNorm): LayerNorm((768,), eps=1e-12, e
      dropout): Dropout(p=0.1, inplace=False)
    )
    encoder): BertEncoder(
      layer): ModuleList(
        0): BertLayer(
          attention): BertAttention(
            query): BertAttention(
              query_weight): BertWeight(
                query_weight): BertWeight(
                  query_weight): BertWeight(
                    query_weight): BertWeight(
                      query_weight): BertWeight(
                        query_weight): BertWeight(
                          query_weight): BertWeight(
                            query_weight): BertWeight(
                              query_weight): BertWeight(
                                query_weight): BertWeight(
                                  query_weight): BertWeight(
                                    query_weight): BertWeight(
                                      query_weight): BertWeight(
                                        query_weight): BertWeight(
                                          query_weight): BertWeight(
                                            query_weight): BertWeight(
                                              query_weight): BertWeight(
                                                query_weight): BertWeight(
                                                  query_weight): BertWeight(
                                                    query_weight): BertWeight(
                                                      query_weight): BertWeight(
                                                        query_weight): BertWeight(
                                                          query_weight): BertWeight(
                                                            query_weight): BertWeight(
                                                              query_weight): BertWeight(
                                                                query_weight): BertWeight(
                                                                  query_weight): BertWeight(
                                                                    query_weight): BertWeight(
                                                                      query_weight): BertWeight(
                                                                        query_weight): BertWeight(
                                                                          query_weight): BertWeight(
                                                                            query_weight): BertWeight(
                                                                              query_weight): BertWeight(
                                                                                query_weight): BertWeight(
                                                                                  query_weight): BertWeight(
                                                                                    query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
                                                                                      query_weight): BertWeight(
................................................................
```

[CLS] what is important to us ? [SEP] it is important to us to include , em ##power and support humans of all kinds. [SEP]

Highly attributed for end position

BertEmbedding

what
?
us
kinds
is



EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
# explaining layers
```

```
BertForQuestionAnswering(  
    bert): BertModel(  
        embeddings): BertEmbeddings(  
            word_embeddings): Embedding(28996, 768, padding_idx=0)  
            position_embeddings): Embedding(512, 768)  
            token_type_embeddings): Embedding(2, 768)  
            LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
            dropout): Dropout(p=0.1, inplace=False)  
        )  
        encoder): BertEncoder(  
            layer): ModuleList(  
                0): BertLayer(  
                    attention): BertAttention(  
                        self): BertSelfAttention(  
                            query): Linear(in_features=768, out_features=768, bias=True)  
                            key): Linear(in_features=768, out_features=768, bias=True)  
                            value): Linear(in_features=768, out_features=768, bias=True)  
                            dropout): Dropout(p=0.1, inplace=False)  
                        )  
                    )  
                    intermediate): BertIntermediate(  
                        dense): Linear(in_features=768, out_features=3072, bias=True)  
                    )  
                    output): BertOutput(  
                        dense): Linear(in_features=3072, out_features=768, bias=True)  
                        LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
                        dropout): Dropout(p=0.1, inplace=False)  
                    )  
                )  
                1): BertLayer(  
                    attention): BertAttention(  
                        self): BertSelfAttention(  
                            query): Linear(in_features=768, out_features=768, bias=True)  
                            key): Linear(in_features=768, out_features=768, bias=True)  
                            value): Linear(in_features=768, out_features=768, bias=True)  
                            dropout): Dropout(p=0.1, inplace=False)  
                        )  
                    )  
                )  
            )  
        )  
    )  
)
```



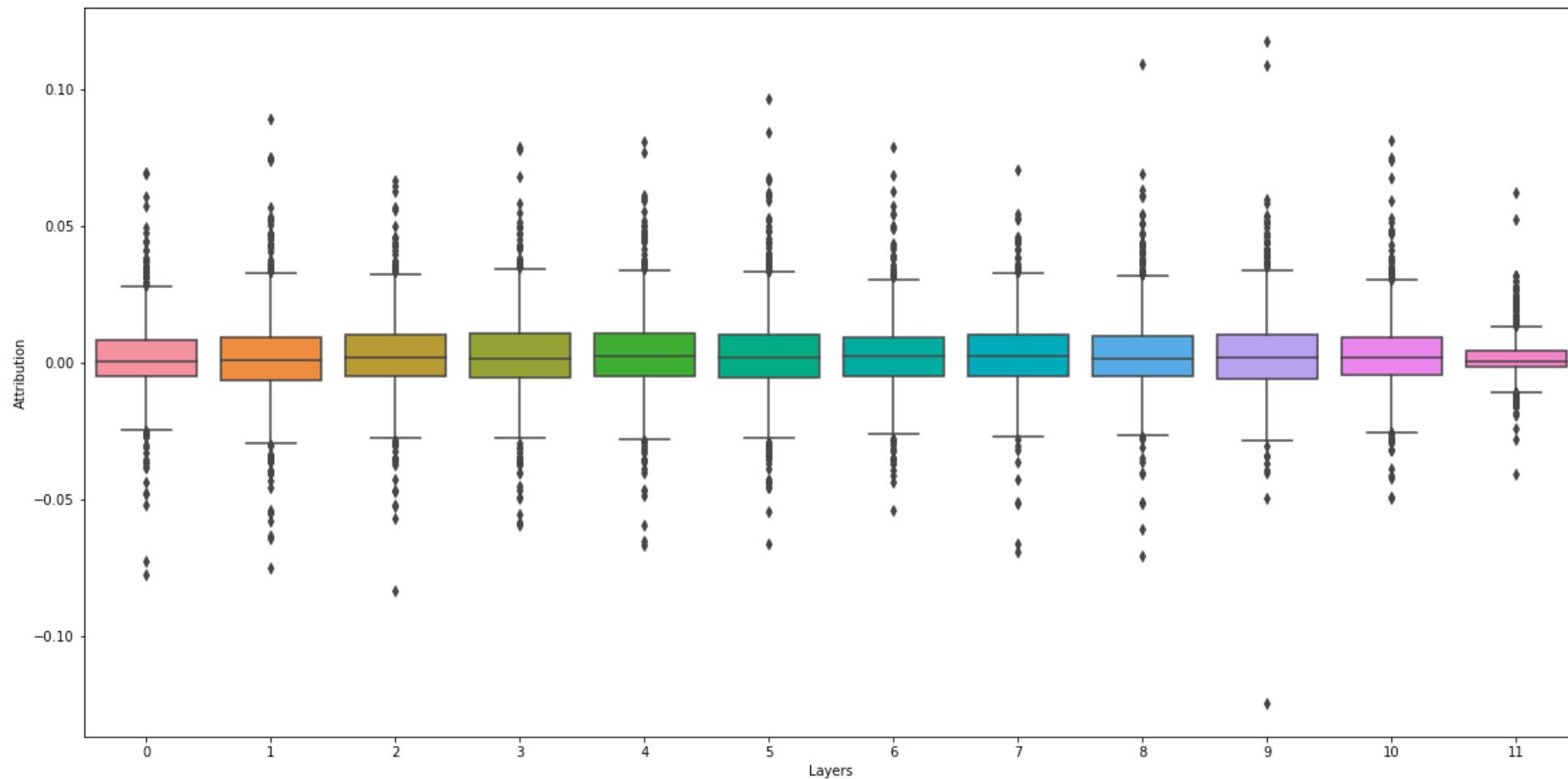
EXPLAINING BERT MODELS FOR QUESTION ANSWERING

```
# explaining layers  
  
for i in range(model.config.num_hidden_layers):  
    lc = LayerConductance(squad_pos_forward_func,  
                           model.bert.encoder.layer[i])  
  
    layer_attributions_start = lc.attribute(  
        input_embed, baselines=ref_emb, ..., 0))  
  
    layer_attributions_end = lc.attribute(  
        input_embed, baselines=ref_emb, ..., 1))
```

```
BertForQuestionAnswering(  
    bert): BertModel(  
        embeddings): BertEmbeddings(  
            word_embeddings): Embedding(28996, 768, padding_idx=0)  
            position_embeddings): Embedding(512, 768)  
            token_type_embeddings): Embedding(2, 768)  
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
            (dropout): Dropout(p=0.1, inplace=False)  
        )  
        encoder): BertEncoder(  
            (layer): ModuleList(  
                (0): BertLayer(  
                    (attention): BertAttention(  
                        (self): BertSelfAttention(  
                            (query): Linear(in_features=768, out_features=768, bias=True)  
                            (key): Linear(in_features=768, out_features=768, bias=True)  
                            (value): Linear(in_features=768, out_features=768, bias=True)  
                            (dropout): Dropout(p=0.1, inplace=False)  
                        )  
                    )  
                    (intermediate): BertIntermediate(  
                        (dense): Linear(in_features=768, out_features=3072, bias=True)  
                    )  
                    (output): BertOutput(  
                        (dense): Linear(in_features=3072, out_features=768, bias=True)  
                        (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)  
                        (dropout): Dropout(p=0.1, inplace=False)  
                    )  
                )  
                (1): BertLayer(  
                    (attention): BertAttention(  
                        (self): BertSelfAttention(  
                            (query): Linear(in_features=768, out_features=768, bias=True)  
                            (key): Linear(in_features=768, out_features=768, bias=True)  
                            (value): Linear(in_features=768, out_features=768, bias=True)  
                            (dropout): Dropout(p=0.1, inplace=False)  
                        )  
                    )  
                )  
            )  
        )  
    )
```

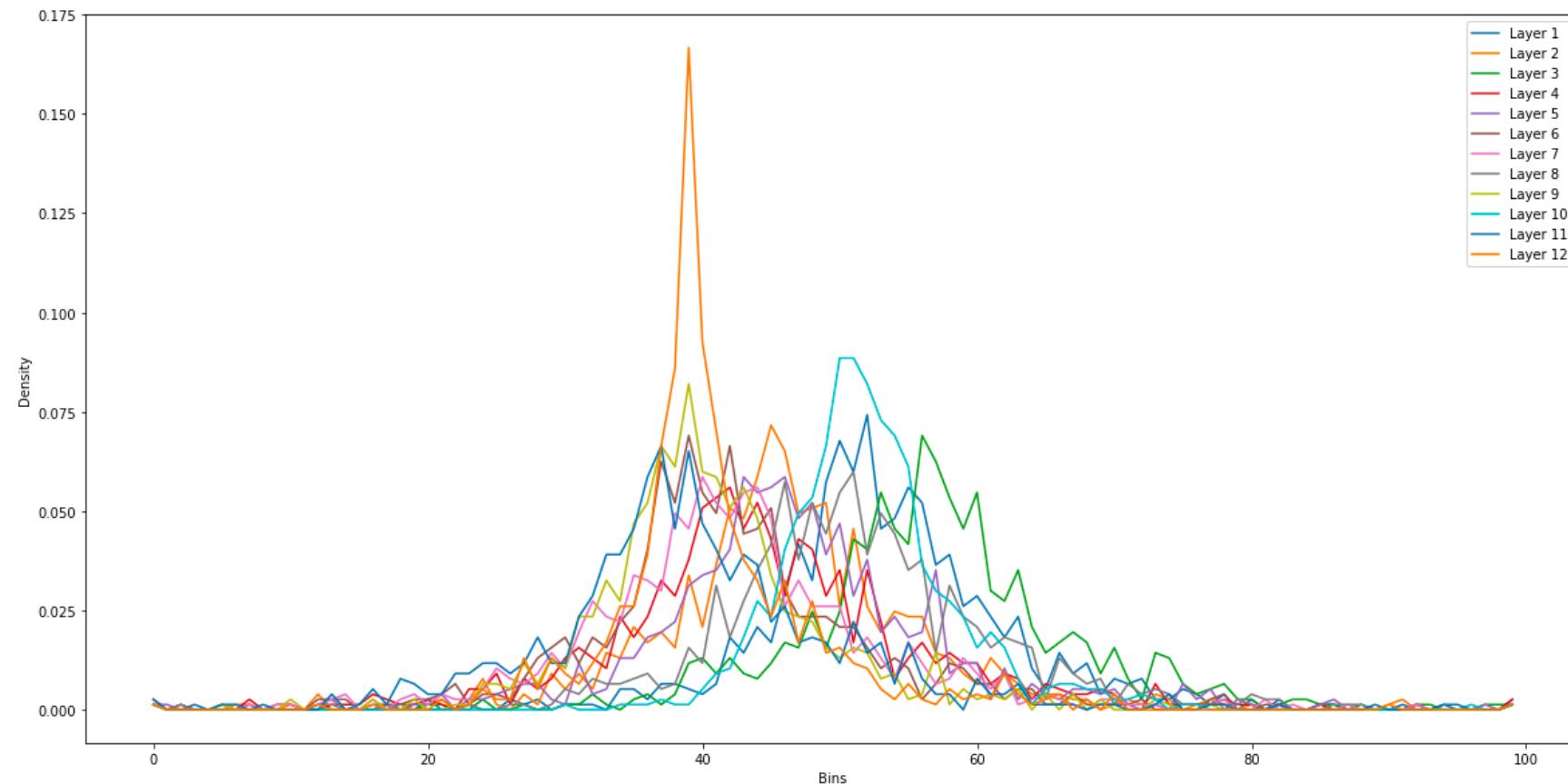


THE DISTRIBUTION OF ATTRIBUTIONS FOR TOKEN `KINDS` IN ALL 12 BERT LAYERS FOR START POSITION PREDICTION



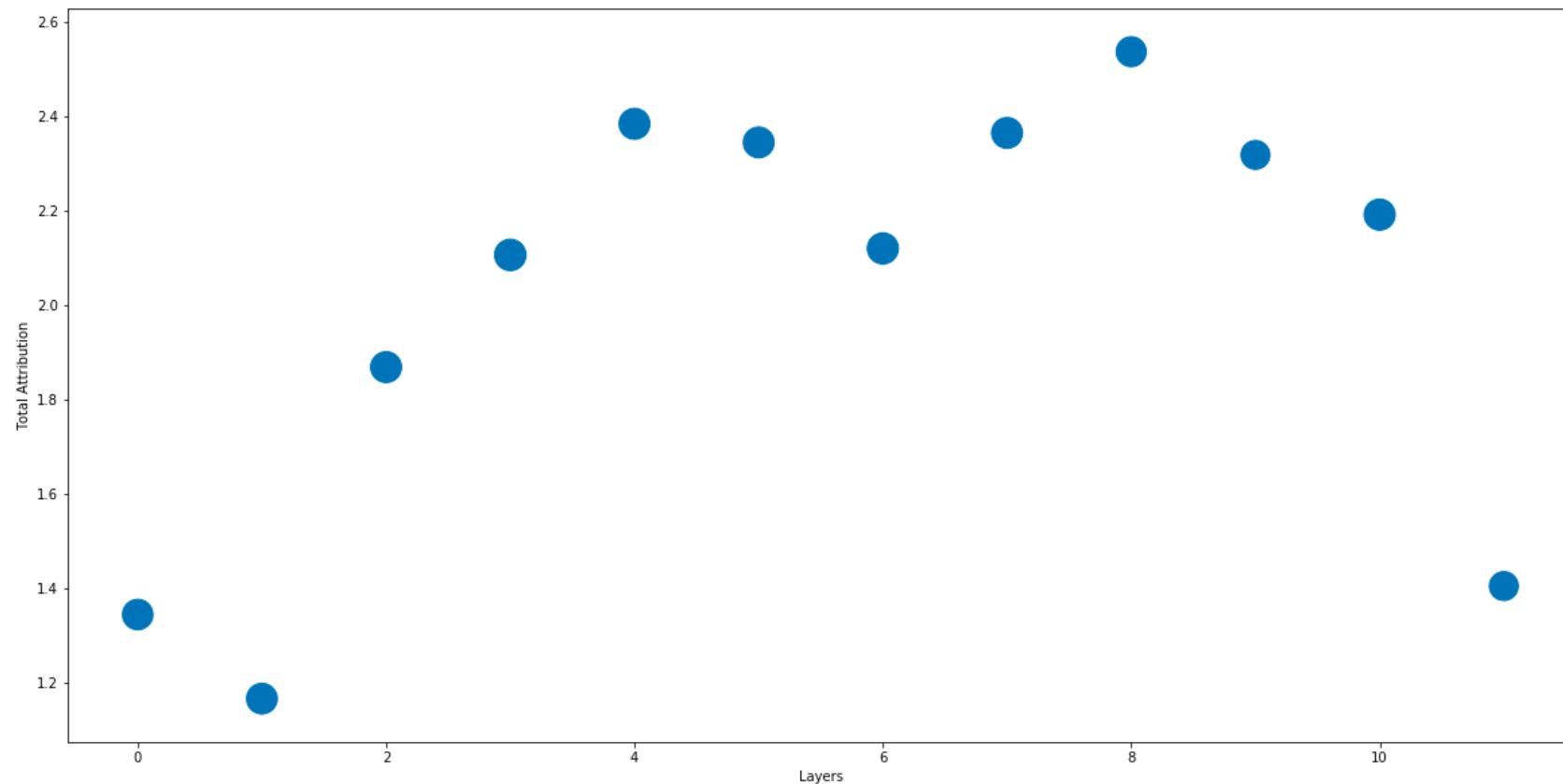


PROBABILITY MASS FUNCTION (PMF) OF ATTRIBUTIONS FOR END POSITION TOKEN `KINDS` PREDICTION



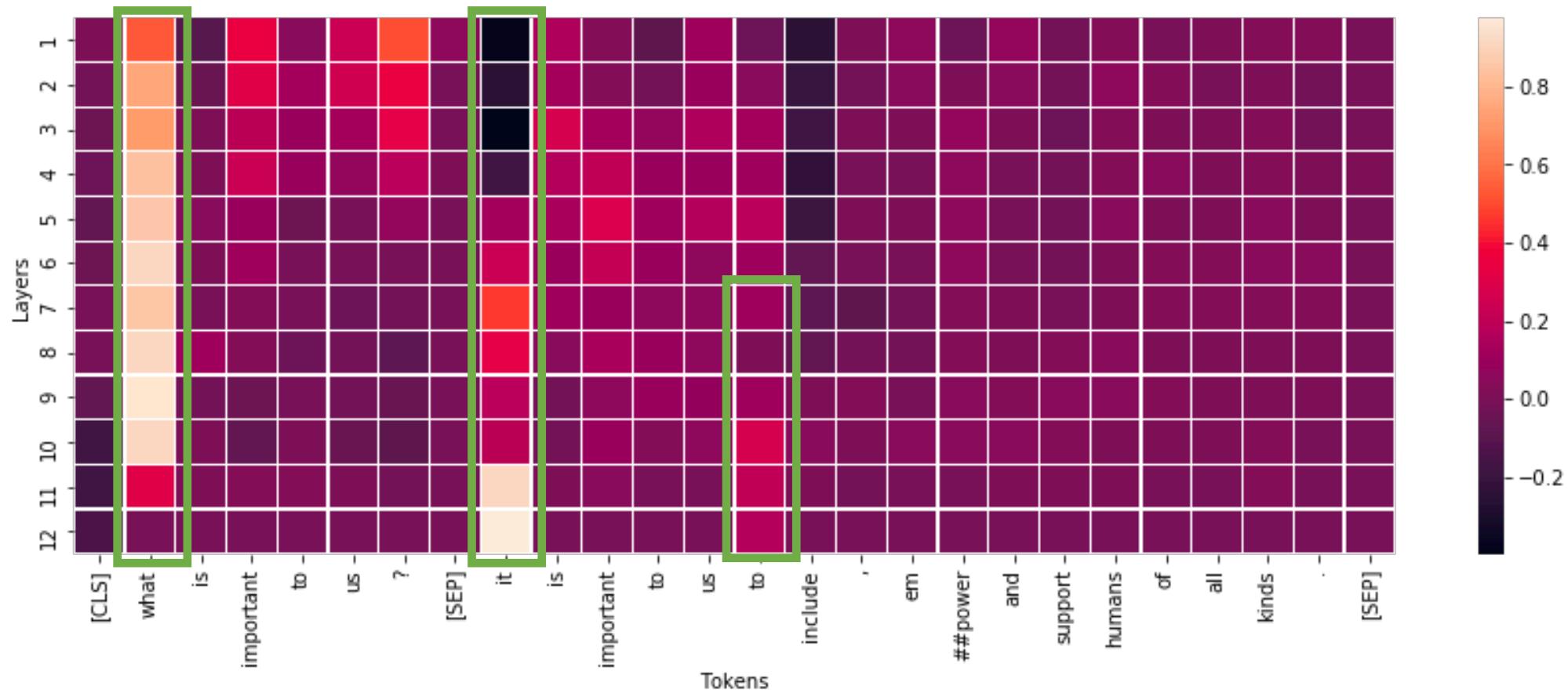


THE ENTROPY OF ATTRIBUTIONS FOR END POSITION TOKEN `KINDS` IN ALL 12 BERT LAYERS



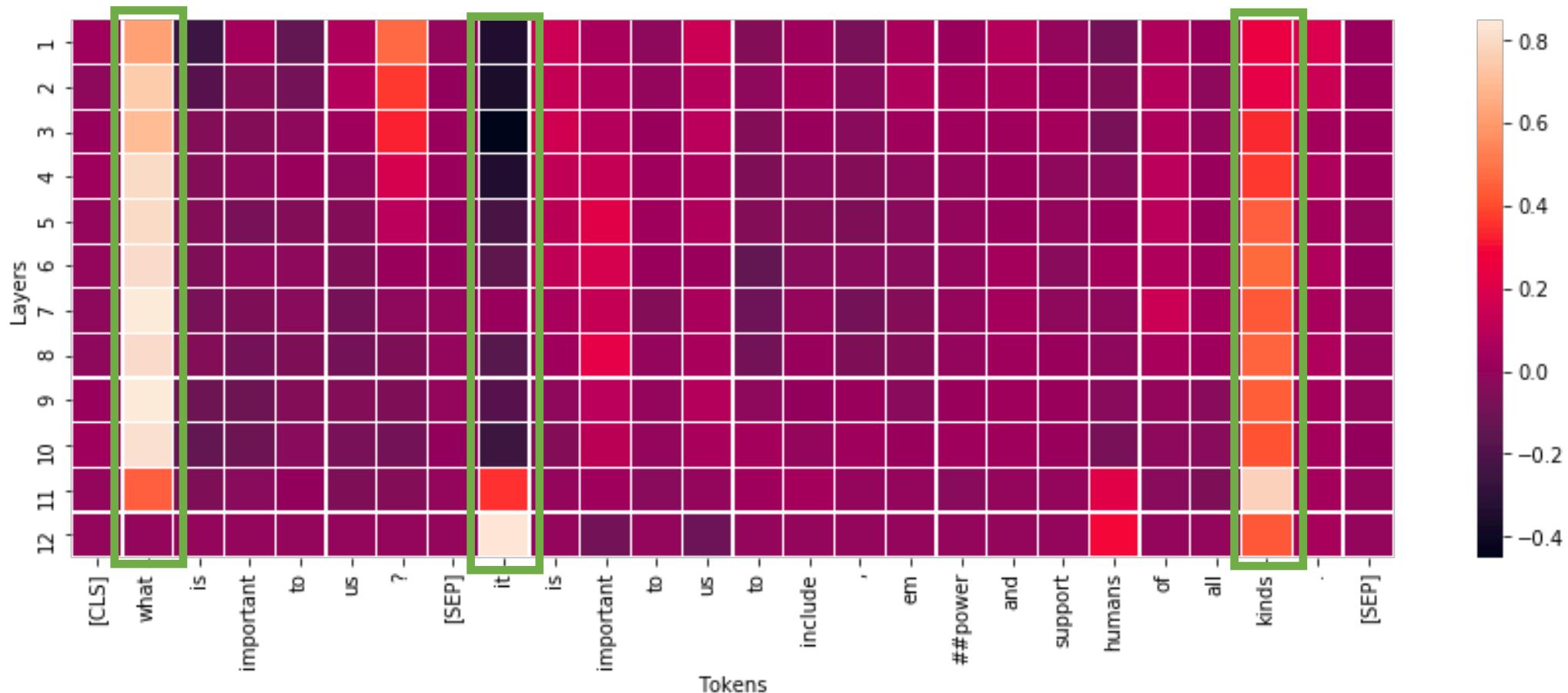


ATTRIBUTION HEAT MAP OF ALL TOKENS ACROSS ALL 12 BERT LAYERS FOR START POSITION PREDICTION





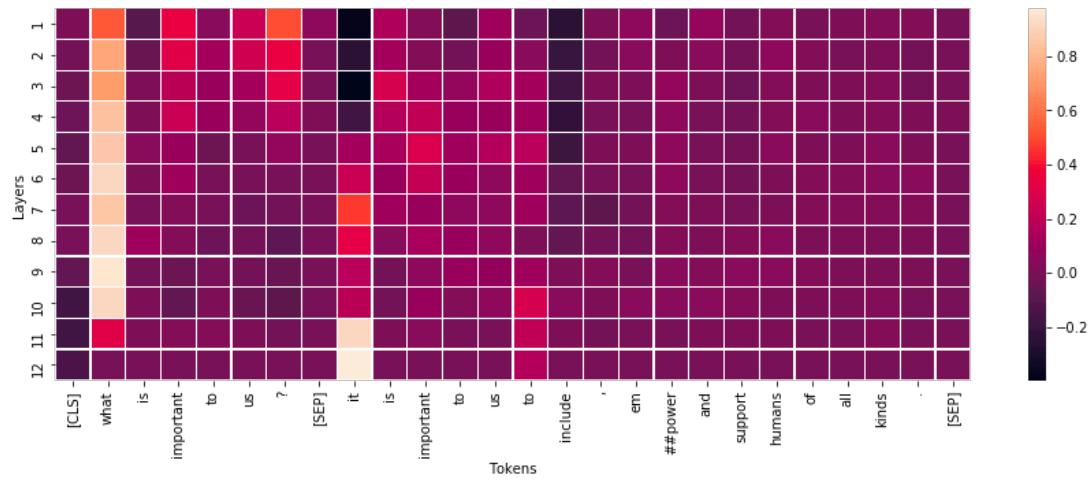
ATTRIBUTION HEAT MAP OF ALL TOKENS ACROSS ALL 12 BERT LAYERS FOR END POSITION PREDICTION





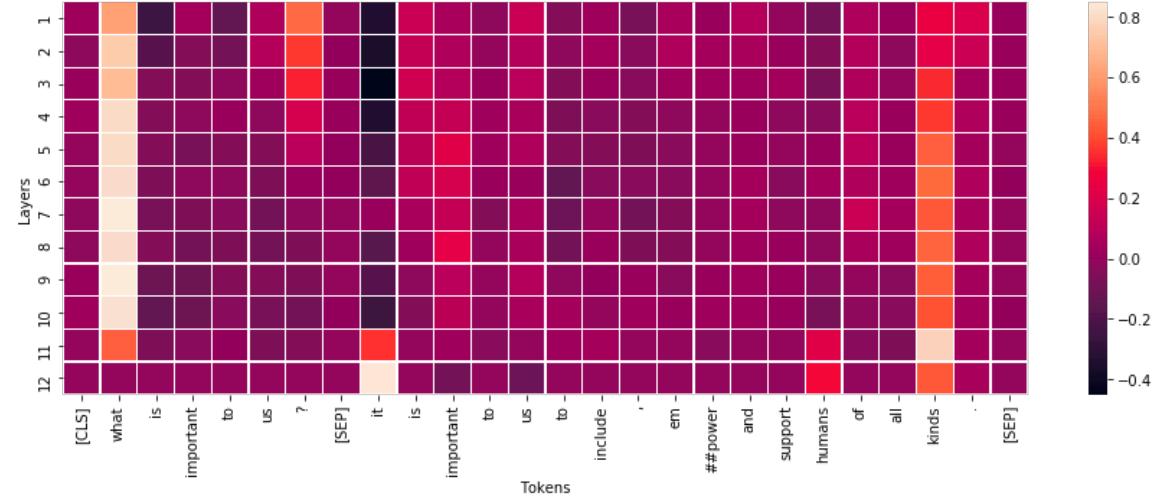
COMPARING HEAT MAPS OF THE START AND END POSITION

Start Position



$P(\text{Start Position}) = 0.52$

End Position



$P(\text{End Position}) = 0.75$



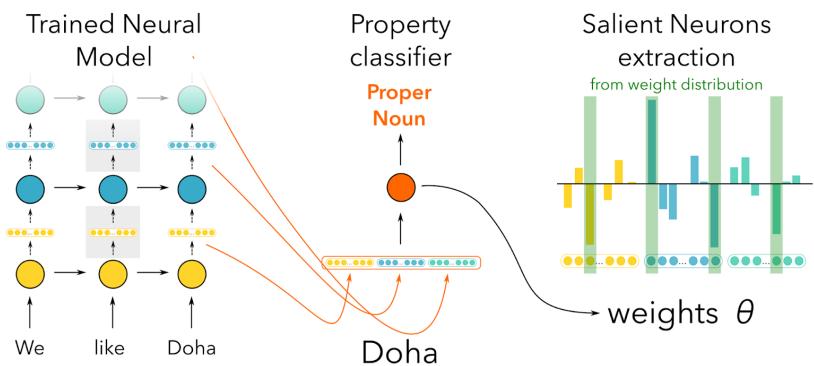
Connecting two worlds

Concept-based neuron analysis
and
Neuron importance attribution for a specific prediction



CONNECTING CONCEPT ANALYSIS AND NEURON CAUSATION ANALYSIS

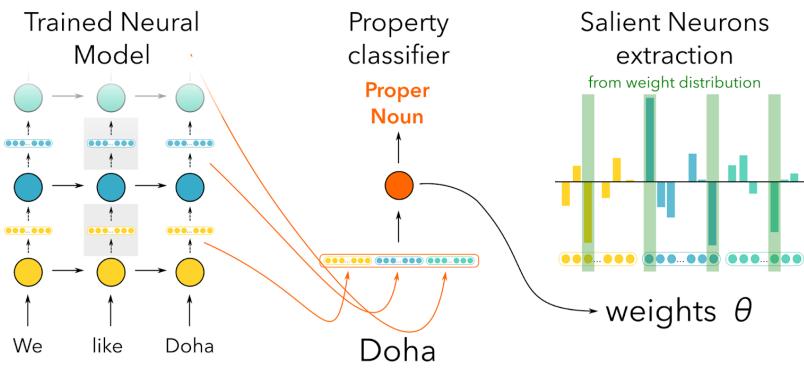
CONCEPT ANALYSIS



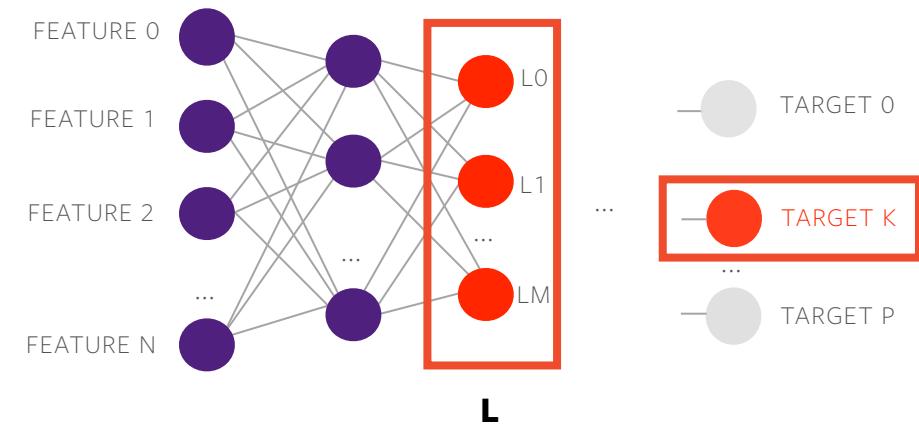


CONNECTING CONCEPT ANALYSIS AND NEURON CAUSATION ANALYSIS

CONCEPT ANALYSIS



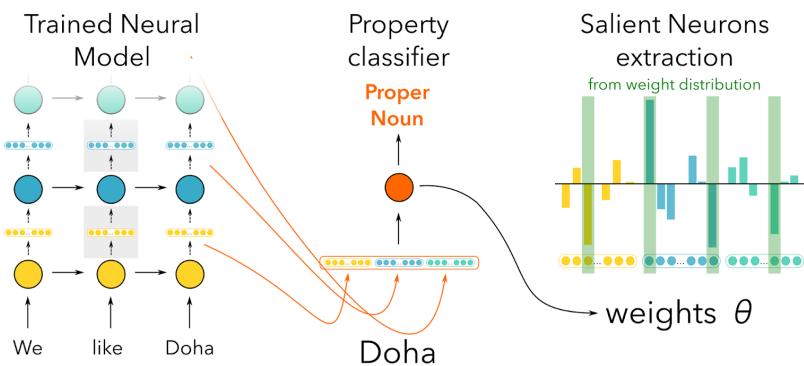
NEURON IMPORTANCE FOR A PREDICTION



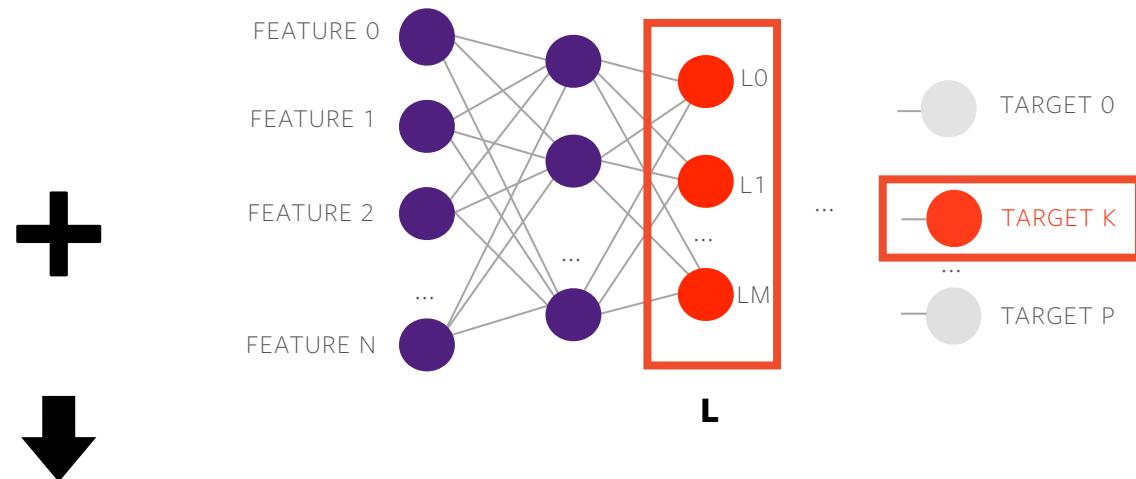


CONNECTING CONCEPT ANALYSIS AND NEURON CAUSATION ANALYSIS

CONCEPT ANALYSIS



NEURON IMPORTANCE FOR A PREDICTION



EXPLAIN A SPECIFIC PREDICTION IN TERMS OF CONCEPTS
WITH HIGH NEURON IMPORTANCE SCORES



QUANTITATIVE TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)



TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Explaining model predictions on the basis of pre-defined concepts

"beautiful" "grandios" "brilliant"
"wonderful" Positive Adjectives

"interesting" "sense" "characteristic"
"laugh" Neutral Terms



TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Explaining model predictions on the basis of pre-defined concepts

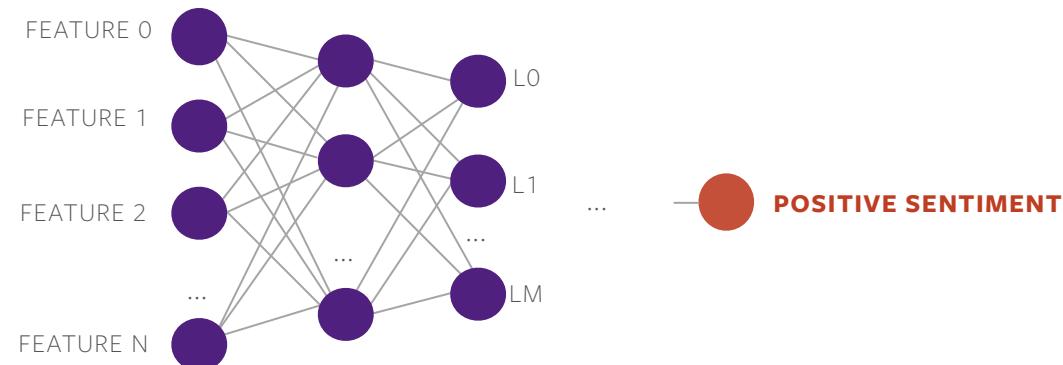
"beautiful" "grandios" "brilliant"
"wonderful" Positive Adjectives

"interesting" "sense" "characteristic"
"laugh" Neutral Terms

- Measure how important high-level concepts are for our predictions



TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)



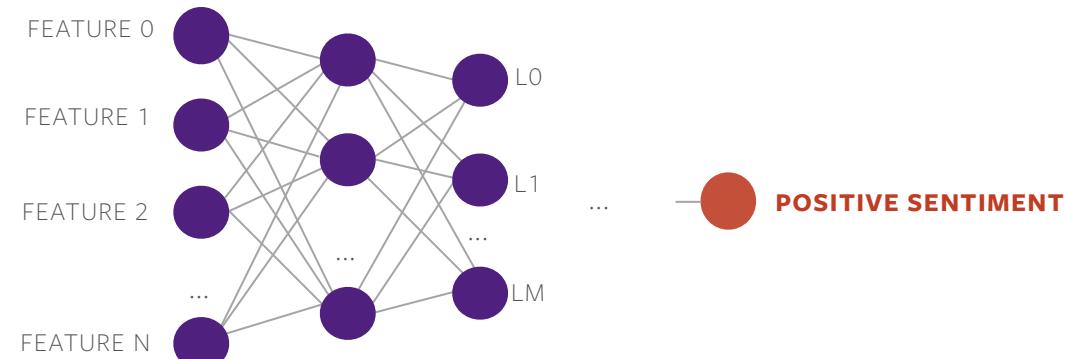


TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

"This is the best film that I've ever seen"

"Last weekend we watched a fantastic performance in the movie theater "

...



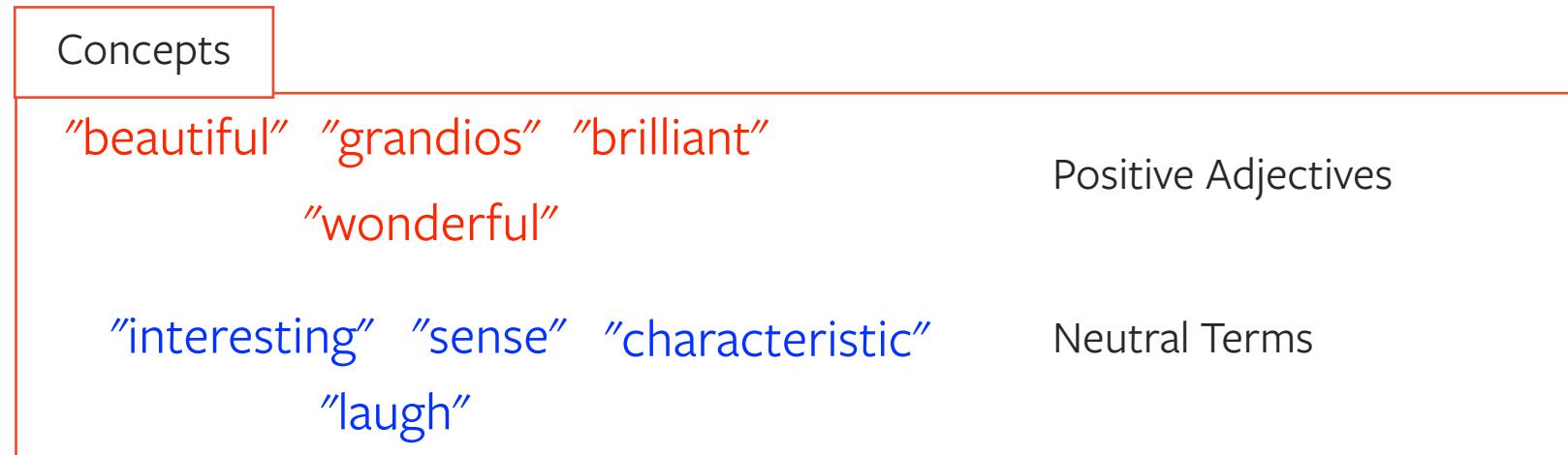


TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

"This is the best film that I've ever seen"

"Last weekend we watched a fantastic performance in the movie theater"

...





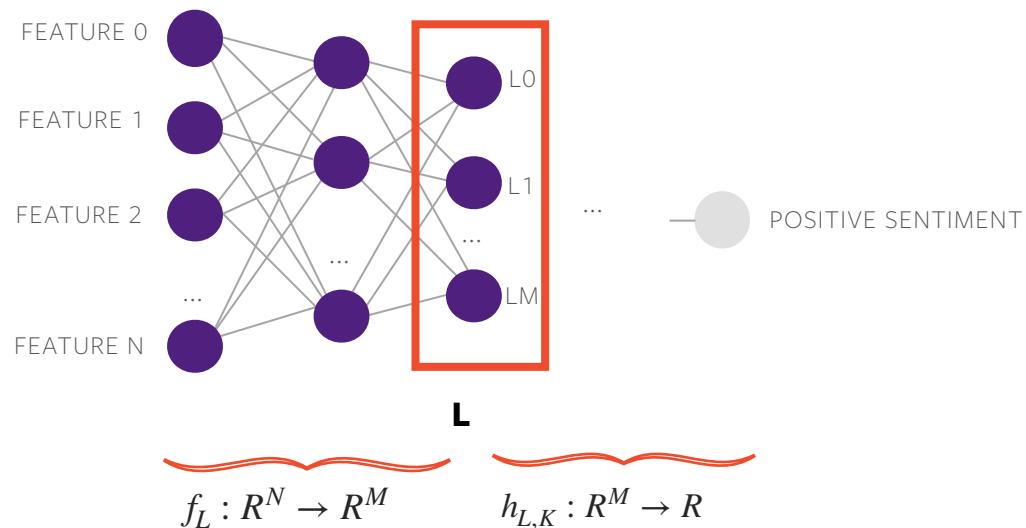
TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Two Step Procedure ([Kim, et.al., 2018](#)) - Estimates concept importance in a DNN layer for model prediction
 - 1) Concept Activation Vector (CAV) Generation
 - 2) Attributions for given test example



TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

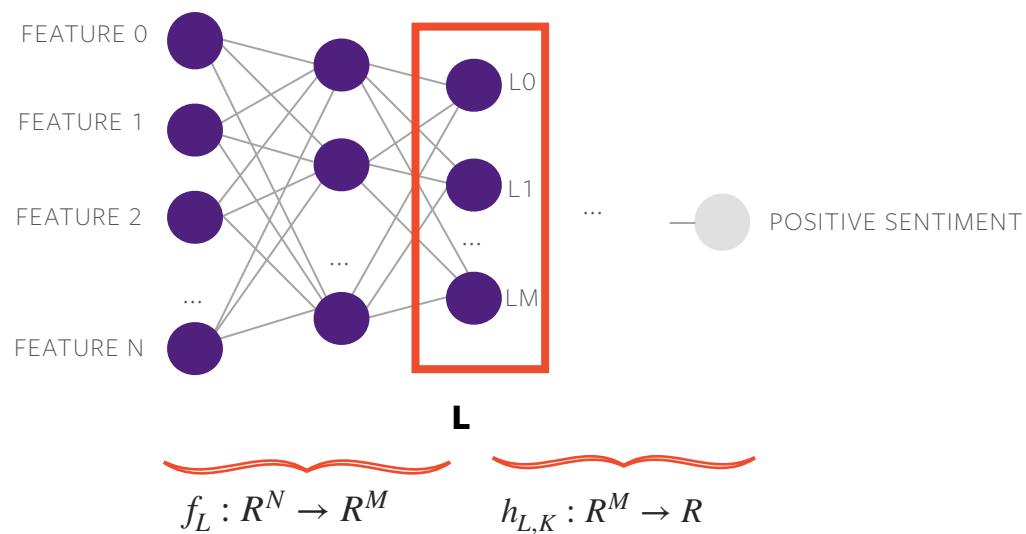
- Two Step Procedure ([Kim, et.al., 2018](#)) - Estimates concept importance in a DNN layer for model prediction
 - 1) Concept Activation Vector (CAV) Generation





TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Two Step Procedure ([Kim, et.al., 2018](#))
 - 1) Concept Activation Vector (CAV) Generation

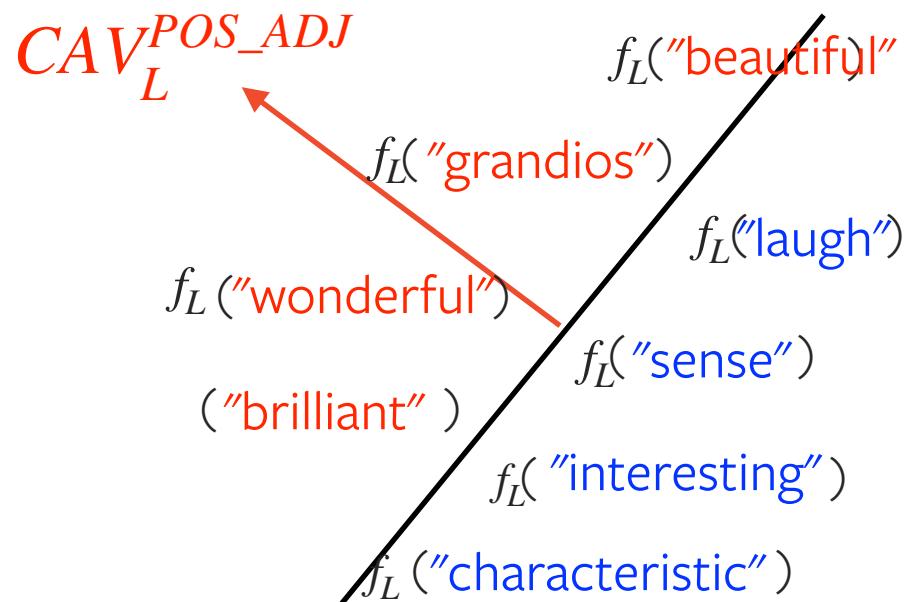
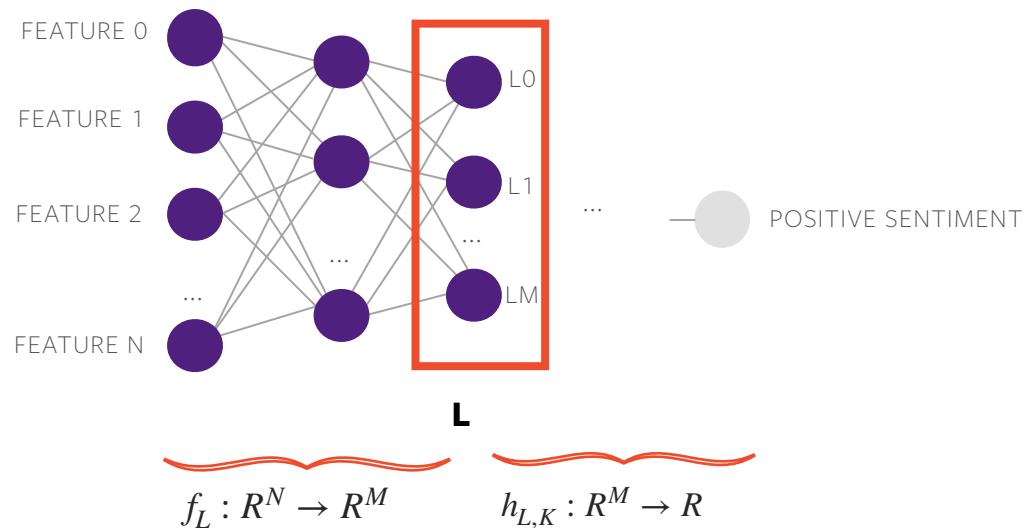


$f_L(\text{"beautiful"})$
 $f_L(\text{"grandios"})$
 $f_L(\text{"laugh"})$
 $f_L(\text{"wonderful"})$
 $f_L(\text{"sense"})$
 $f_L(\text{"brilliant"})$
 $f_L(\text{"interesting"})$
 $f_L(\text{"characteristic"})$



TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Two Step Procedure ([Kim, et.al., 2018](#))
 - 1) Concept Activation Vector (CAV) Generation

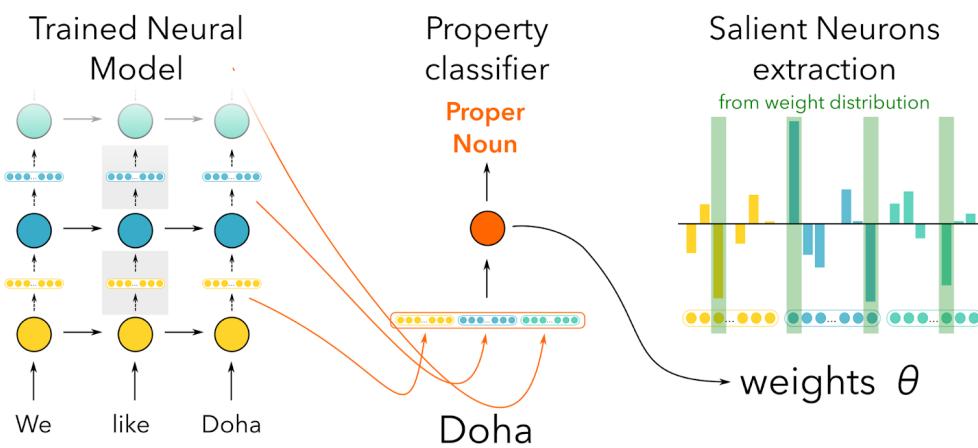


CAV is the vector orthogonal to the hyperplane of concept linear classifier

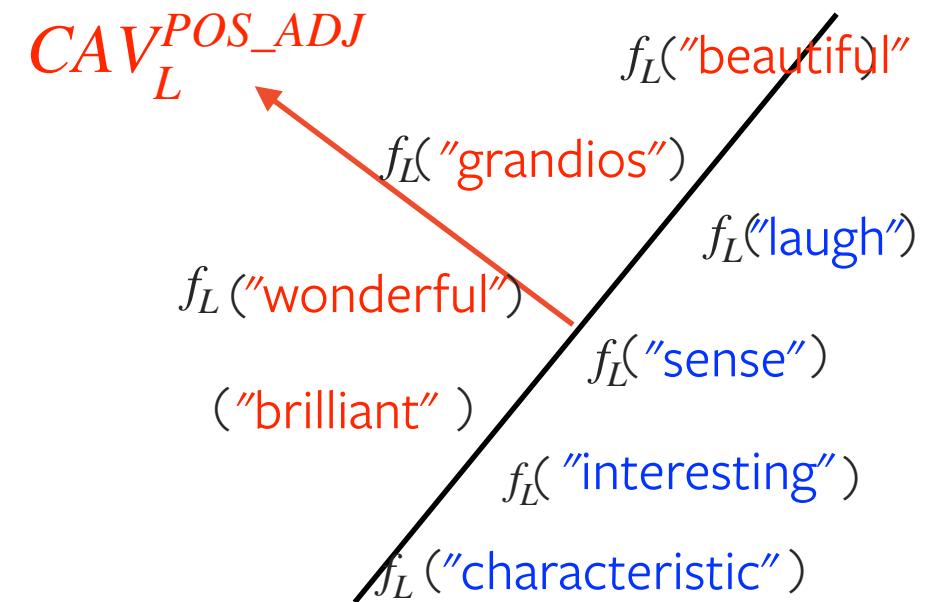


ANALOGY BETWEEN LINEAR CLASSIFIER IN NEURON PROBING AND CAV

NEURON PROBING BY A LINEAR CLASSIFIER



LEARNING CONCEPT ACTIVATION VECTORS BY A LINEAR CLASSIFIER

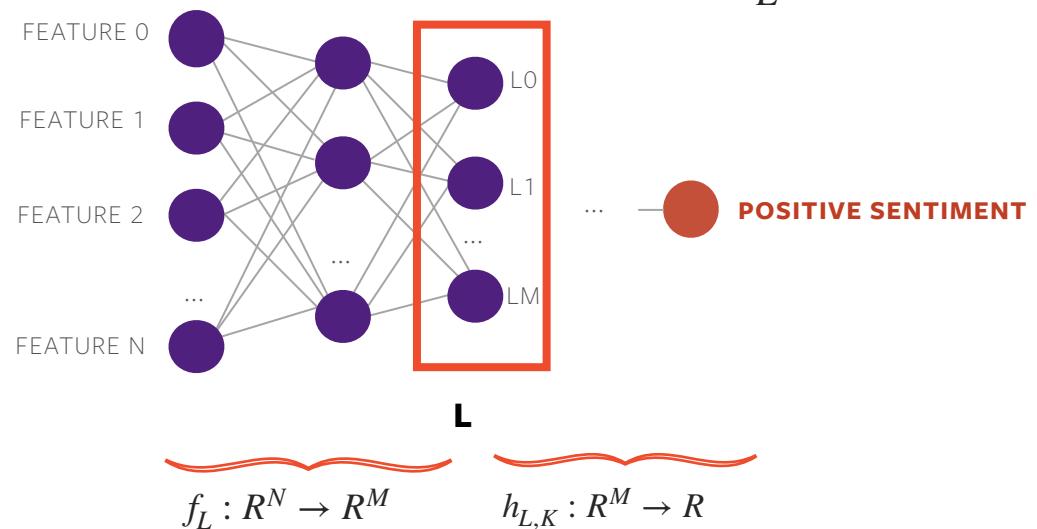




TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Two Step Procedure ([Kim, et.al., 2018](#))
 - 1) Concept Activation Vector (CAV) Generation

2) Attributions for given test example



$$ATTR_L^{POSITIVE_CLASS}(\text{"This is the best film ..."}, \text{"Last weekend we watched a fantastic..."}) = \frac{\partial h_{L,K}(f_L(\text{"This is the best film ..."}, \text{"Last weekend we watched a fantastic..."}))}{\partial f_L(\text{"This is the best film ..."}, \text{"Last weekend we watched a fantastic..."})}$$



TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Two Step Procedure ([Kim, et.al., 2018](#))
 - 1) Concept Activation Vector (CAV) Generation
 - 2) Attributions for given test example

$$CAV_ATTR_{L, POSITIVE_CLASS}^{POS_ADJ}(\text{ "This is the best film ..."} \\ \text{ "Last weekend we watched } \\ \text{ a fantastic... " }) = CAV_L^{POS_ADJ} \cdot ATTR_L^{POSITIVE_CLASS}(\text{ "This is the best film ..."} \\ \text{ "Last weekend we watched } \\ \text{ a fantastic... " })$$



TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Two Step Procedure ([Kim, et.al., 2018](#))
 - 1) Concept Activation Vector (CAV) Generation
 - 2) Attributions for given test example

$$CAV_ATTR_{L,POSITIVE_CLASS}^{POS_ADJ} \left(\begin{array}{c} \text{"This is the best film ..."} \\ \text{"Last weekend we watched} \\ \text{a fantastic..."} \end{array} \right)$$

$$= CAV_L^{POS_ADJ} \cdot ATTR_L^{POSITIVE_CLASS} \left(\begin{array}{c} \text{"This is the best film ..."} \\ \text{"Last weekend we watched} \\ \text{a fantastic..."} \end{array} \right)$$

$$TCAV_{L,POSITIVE_CLASS}^{POS_ADJ} \left(\begin{array}{c} \text{"This is the best film ..."} \\ \text{"Last weekend we watched} \\ \text{a fantastic..."} \end{array} \right) =$$

$$\frac{\left| CAV_ATTR_{L,POSITIVE_CLASS}^{POS_ADJ} \left(\begin{array}{c} \text{"This is the best film ..."} \\ \text{"Last weekend we watched} \\ \text{a fantastic..."} \end{array} \right) \right| > 0}{\left| \begin{array}{c} \text{"This is the best film ..."} \\ \text{"Last weekend we watched} \\ \text{a fantastic..."} \end{array} \right|}$$



TESTING WITH CONCEPT ACTIVATION VECTORS (TCAV)

- Two Step Procedure ([Kim, et.al., 2018](#))
 - 1) Concept Activation Vector (CAV) Generation
 - 2) Attributions for given test example

In a general case

$$TCAV_{L,CLASS}^{CONCEPT}(inputs_{CLASS}) = \frac{|CAV_ATTR_{L,CLASS}^{CONCEPT}(inputs_{CLASS}) > 0|}{|inputs_{CLASS}|}$$



TCAV > STATISTICAL SIGNIFICANCE TESTING

- TCAV can potentially learn meaningless CAVs for a meaningful concept
- A CAV generated for a random concept can potentially be meaningful



TCAV > STATISTICAL SIGNIFICANCE TESTING

- TCAV can potentially learn meaningless CAVs for a meaningful concept
- A CAV generated for a random concept can potentially be meaningful
- Steps we can take to mitigate those issues
 - Two sided statistical significance tests
 - Against large number of neural concepts
 - A meaningful concept will stand out with high TCAV score among most neutral concepts



TCAV: LIMITATIONS

- Concepts has to be pre-defined in advance
 - Time consuming process
 - Learning meaningless CAVs
 - Statistical significance testing for multiple neutral concepts
 - Computationally time and memory intensive



A CASE STUDY ON SENTIMENT ANALYSIS USING TCAV

- Trained a CNN model using IMDB dataset
https://captum.ai/tutorials/IMDB_TorchText_Interpret
- Explaining hand-picked examples of movie rating text descriptions

⌚

```
predict = model( )
```

"This is the best film ..."

"Last weekend we watched
a fantastic... "

⌚

`predict = model(`

 "This is the best film ..."
 >Last weekend we watched
 a fantastic... "

concepts

vs

"beautiful" "interesting"
"grandios" "sense"

pos_adj neutral

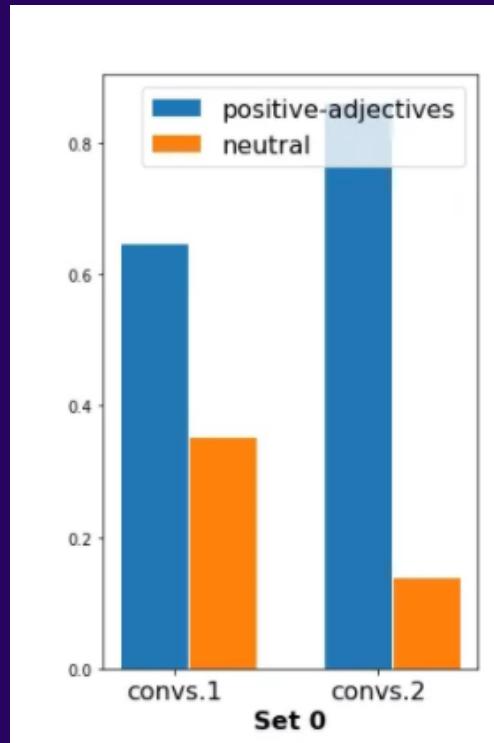


`predict = model(`

"This is the best film ..."
"Last weekend we watched
a fantastic... "

Sensitivity of model prediction to pos_adj and neutral concept

TCAV Score



Layers

concepts
vs
"beautiful" "interesting"
"grandios" "sense"
pos_adj neutral



`predict = model(`

"This is the best film ..."
"Last weekend we watched
a fantastic... "

concepts
vs

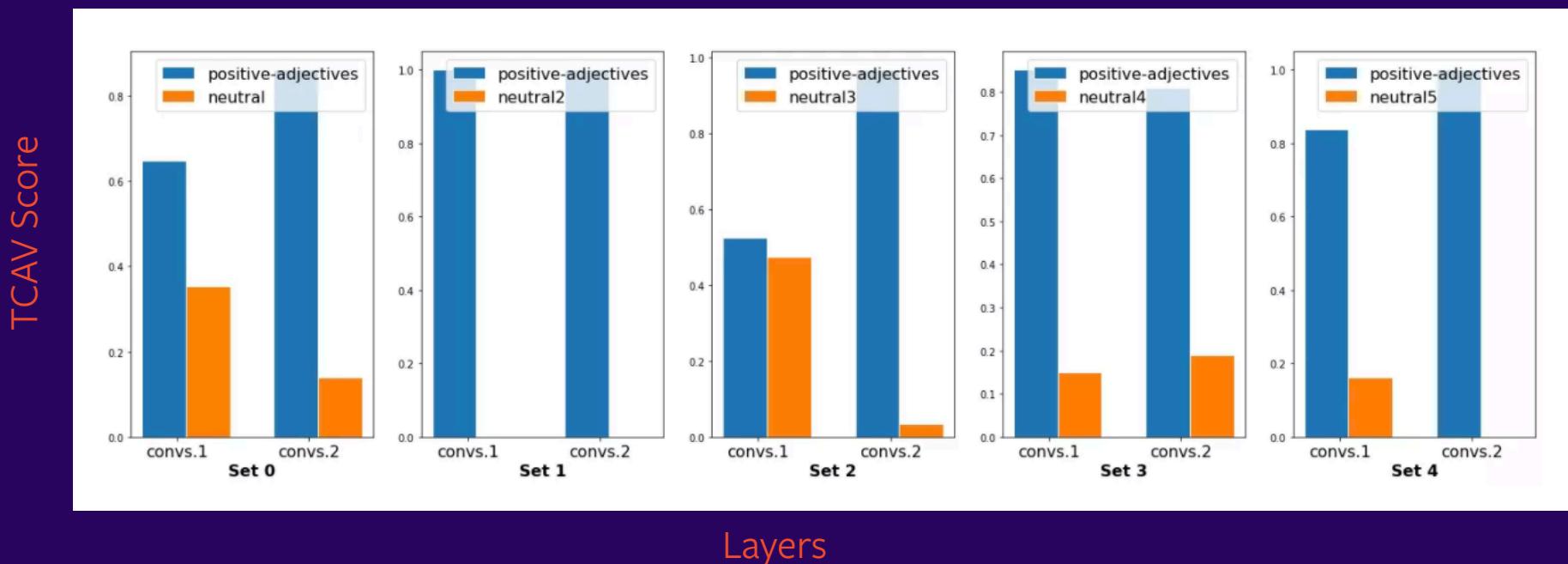
"beautiful"
"grandios"

"interesting"
"sense"

pos_adj

neutral

Sensitivity of model prediction to pos_adj and neutral concept





`predict = model(`

"This is the best film ..."
"Last weekend we watched
a fantastic... "

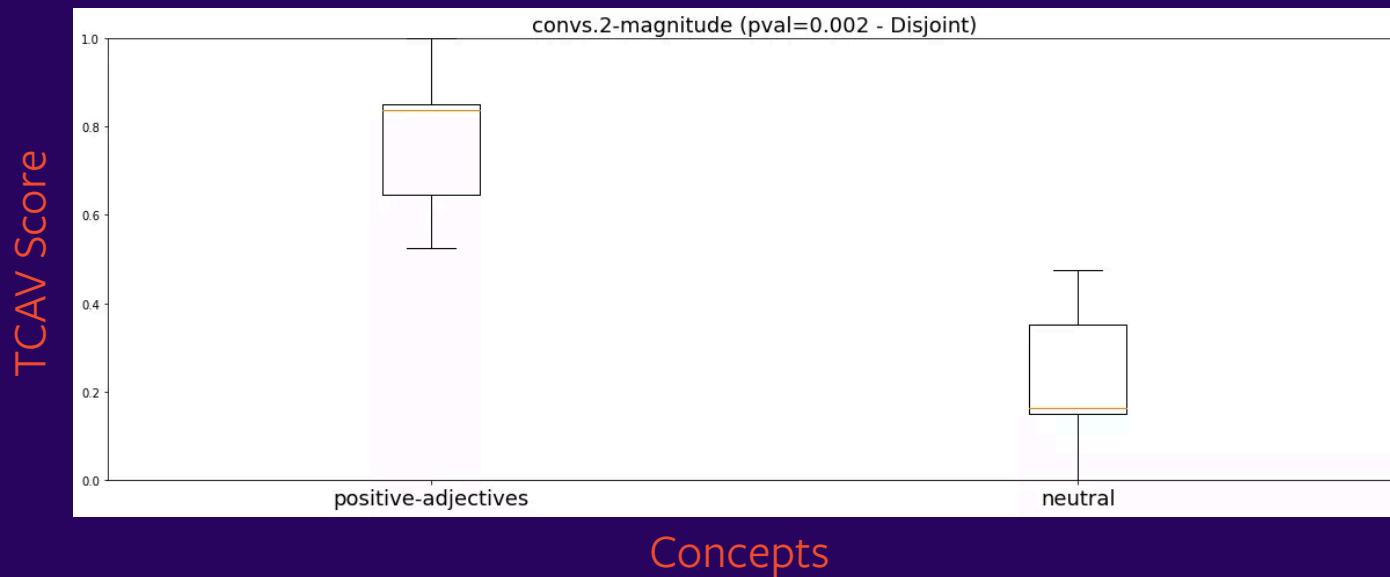
concepts

vs

"beautiful"
"grandios" | "interesting"
"sense"

pos_adj | neutral

The distribution of TCAV scores across multiple pos_adj - neutral experiments in conv2 layer





`predict = model(`

"This is the best film ..."
"Last weekend we watched
a fantastic... "

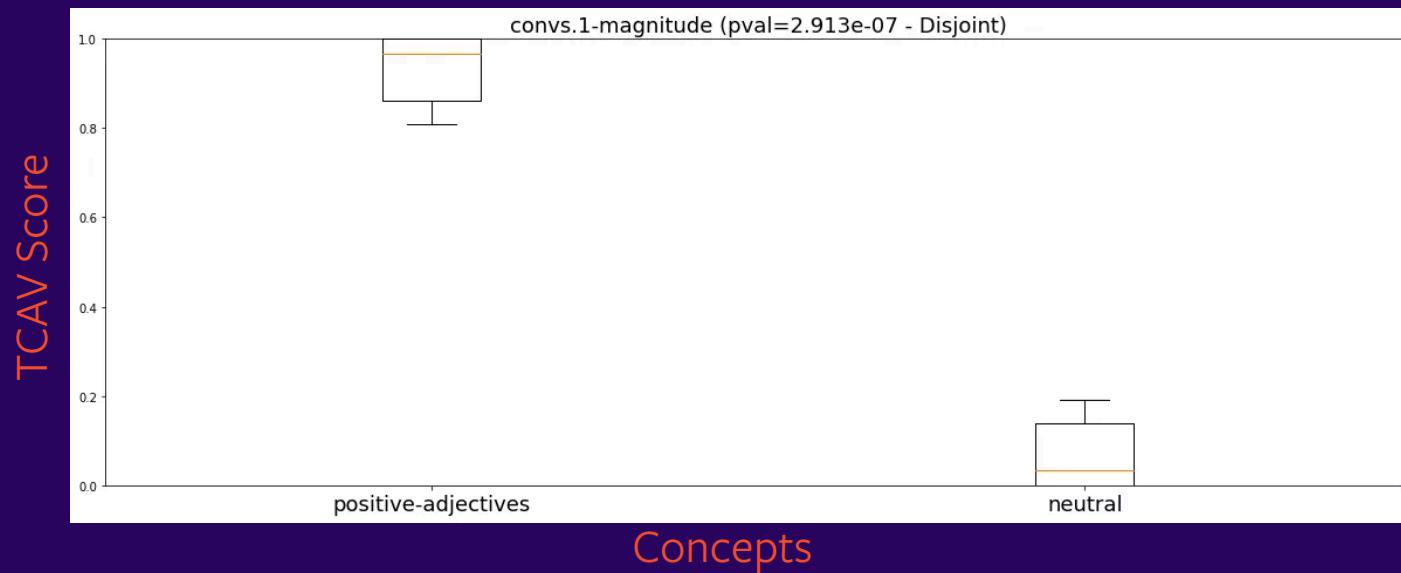
concepts

vs

"beautiful"
"grandios" | "interesting"
"sense"

pos_adj | neutral

The distribution of TCAV scores across multiple pos_adj - neutral experiments in conv1 layer





MORE TECHNIQUES ON CONCEPT-BASED MODEL INTERPRETABILITY

- Automatic Concept Extraction (ACE) for images ([Towards Automatic Concept-based Explanations, Ghorbani et. al., 2019](#))
- Identifying a sufficient set of concepts that describe our prediction, ConceptSHAP ([On Completeness-aware Concept-Based Explanations in Deep Neural Networks, Yeh, et. al., 2020](#))

Summary and Discussion

Thank you
