

Interpretability and Analysis in Neural NLP

Yonatan Belinkov, Sebastian Gehrmann, Ellie Pavlick

ACL Tutorial
July 5, 2020

Analysis Questionnaire

What is the goal of the study?

Pedagogical / Debugging / Debiasing / ...

Understanding model structure / model decisions / data / ...

How do you quantify an outcome?

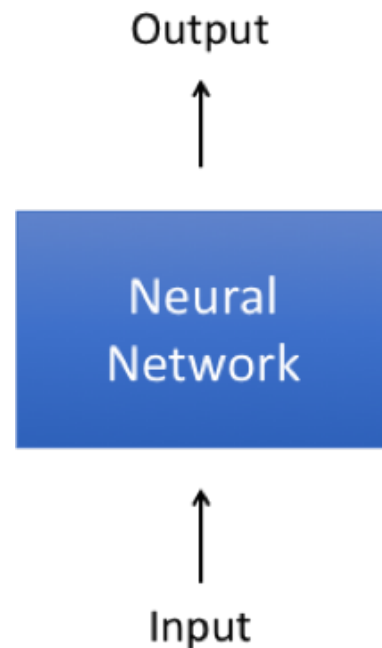
Who is your user or target group?

ML or NLP Expert/ Domain Expert / Student / Lay User of the System ...

How much domain/ model knowledge do they have?

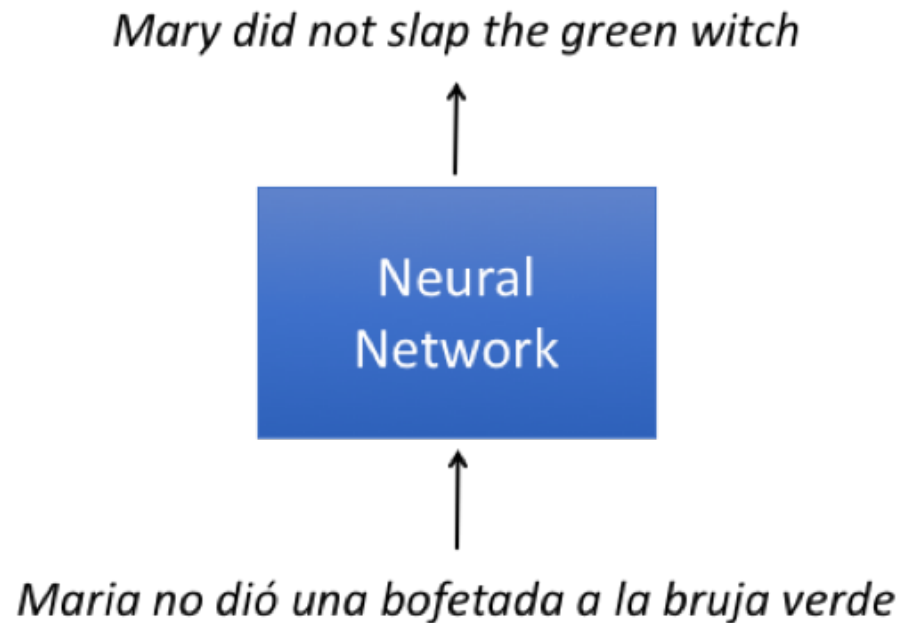
End-to-End Learning

- The predominant approach in NLP these days is end-to-end learning
- Learn a model $f : x \rightarrow y$, which maps input x to output y



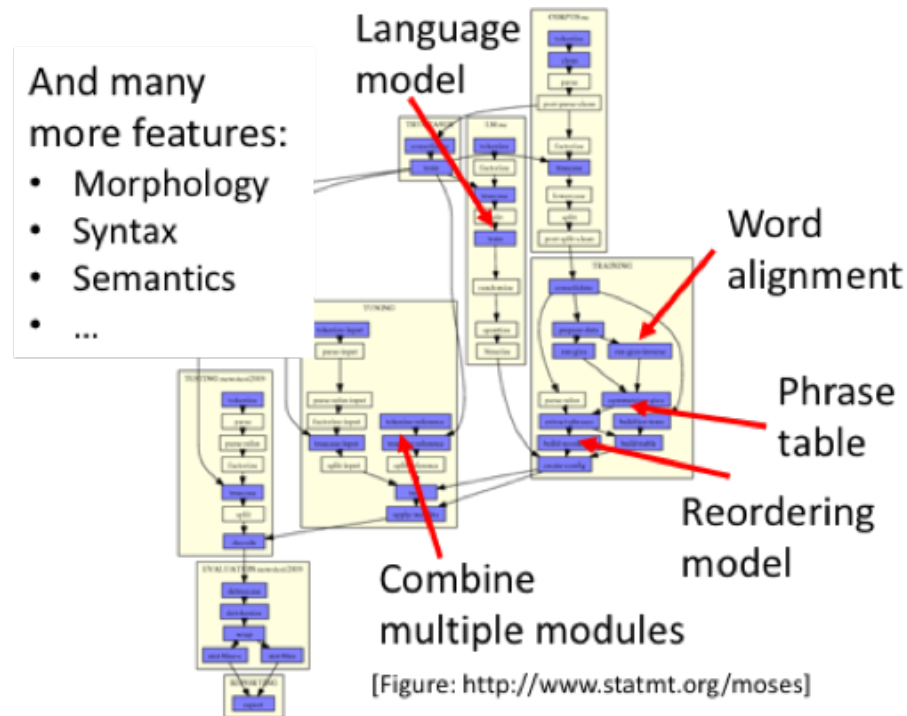
End-to-End Learning

- For example, in machine translation we map a source sentence to a target sentence, via a deep neural network:



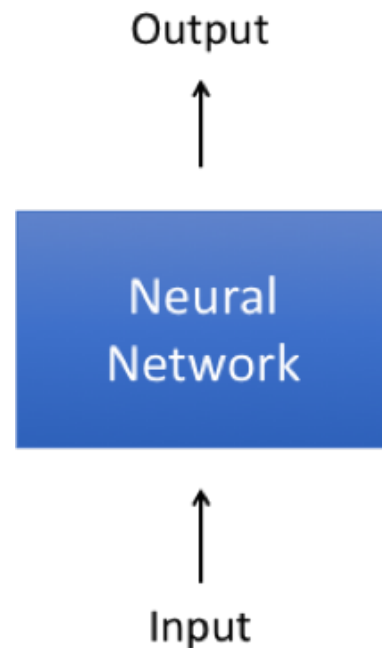
A Historical Perspective

- Compare this with a traditional statistical approach to MT, based on multiple modules and features:



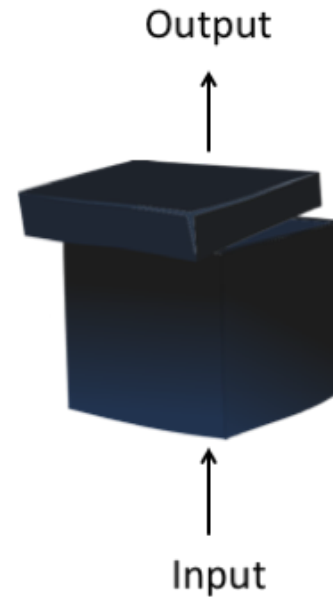
End-to-End Learning

- The predominant approach in NLP these days is end-to-end learning, where all parts of the model are trained on the same task:



How can we open the black box?

- Given $f : x \rightarrow y$, we want to ask some questions about f
 - What is its internal structure?
 - How does it behave on different data?
 - Why does it make certain decisions?
 - When does it succeed/fail?
 - ...



Why should we care?

- Much deep learning research:
 - Trial-and-error, shot in the dark
 - Better understanding → better systems



Why should we care?

- Much deep learning research:
 - Trial-and-error, shot in the dark
 - Better understanding → better systems
- Accountability, trust, and bias in machine learning
 - “Right to explanation”, EU regulation
 - Life threatening situations: healthcare, autonomous cars
 - Better understanding → more accountable systems



Why should we care?

- Much deep learning research:
 - Trial-and-error, shot in the dark
 - Better understanding → better systems
- Accountability, trust, and bias in machine learning
 - “Right to explanation”, EU regulation
 - Life threatening situations: healthcare, autonomous cars
 - Better understanding → more accountable systems
- Neural networks aid the scientific study of language ([Linzen 2019](#))
 - Models of human language acquisition
 - Models of human language processing
 - Better understanding → more interpretable models



Outline

- Structural analyses
- Behavioral analyses
- Interactive visualizations
- Other methods

Outline

- **Structural analyses**
- Behavioral analyses
- Interactive visualizations
- Other methods

Structural Analyses

- Let $f : x \rightarrow y$ be a model mapping an input x to an output y
 - f might be a complicated neural network with many layers or other components
 - For example, $f_l(x)$ might be the output of the network at the l -th layer
- Some questions we might want to ask:
 - What is the role of different components of f ?
 - What kind of information do different components capture?
 - More specifically: Does components A know something about property B?

Structural Analyses

- Let $f : x \rightarrow y$ be a model mapping an input x to an output y
 - f might be a complicated neural network with many layers or other components
 - For example, $f(x)$ might be the output of the network at the l -th layer

Structural Analyses

- Let $f : x \rightarrow y$ be a model mapping an input x to an output y
 - f might be a complicated neural network with many layers or other components
 - For example, $f^l(x)$ might be the output of the network at the l -th layer
- Analysis via a probing classifier
 - Assume a corpus of inputs x with linguistic annotations z
 - Generate representations of x from some part of the model f , for example representations $f^l(x)$ at a certain layer
 - Train another classifier $g : f^l(x) \rightarrow z$ that maps the representations $f^l(x)$ to the property z
 - Evaluate the accuracy of g as a proxy to the quality of representations $f^l(x)$ w.r.t property z

Structural Analyses

- Let $f : x \rightarrow y$ be a model mapping an input x to an output y
 - f might be a complicated neural network with many layers or other components
 - For example, $f^l(x)$ might be the output of the network at the l -th layer
- Analysis via a probing classifier
 - Assume a corpus of inputs x with linguistic annotations z
 - Generate representations of x from some part of the model f , for example representations $f^l(x)$ at a certain layer
 - Train another classifier $g : f^l(x) \rightarrow z$ that maps the representations $f^l(x)$ to the property z
 - Evaluate the accuracy of g as a proxy to the quality of representations $f^l(x)$ w.r.t property z
- In information theoretic terms:
 - Set $h = f(x)$ and recall that $I(h; z) = H(z) - H(z | h)$
 - Then the probing classifier minimizes $H(z | h)$, or maximizes $I(h, z)$

Milestones (partial list)

	f	x	y	g	z
Köhn 2015	Word embedding	Word	Word	Linear	POS, morphology
Ettinger et al. 2016	Sentence embedding	Word, sentence	Word, sentence	Linear	Semantic roles, scope
Shi et al. 2016	RNN MT	Word, sentence	Word, sentence	Linear / tree decoder	Syntactic features, tree
Adi et al. 2017 Conneau et al. 2018	Sentence embedding	sentence	sentence	Linear, MLP	surface, syntax, semantics
Hupkes et al. 2018	RNN, treeRNN	<i>five plus free</i>	<i>eight</i>	Linear	Position, cumulative value
Hewitt+Manning 2019	ELMo, BERT	Sentence	Sentence	Linear	Ful tree

Example Results

- Numerous papers using this methodology to study:
 - Linguistic phenomena (z): phonology, morphology, syntax, semantics
 - Network components (f): word embeddings, sentence embeddings, hidden states, attention weights, etc.
- We'll show example results on machine translation
- Much more related work reviewed in our survey ([Belinkov and Glass 2019](#))

Example: Machine Translation

- Setup
 - f : an RNN encoder-decoder MT model
 - x and y are source and target sentences (lists of words)
 - g : a non-linear classifier (MLP with one hidden layer)
 - z : linguistic properties of words in x or y

Example: Machine Translation

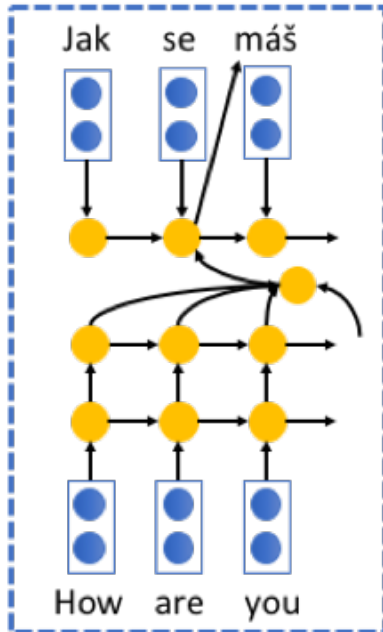
- Setup
 - f : an RNN encoder-decoder MT model
 - x and y are source and target sentences (lists of words)
 - g : a non-linear classifier (MLP with one hidden layer)
 - z : linguistic properties of words in x or y
- Morphology:
 - A challenge for machine translation, previously solved with feature-rich approaches.
 - Do neural networks acquire morphological knowledge?

Example: Machine Translation

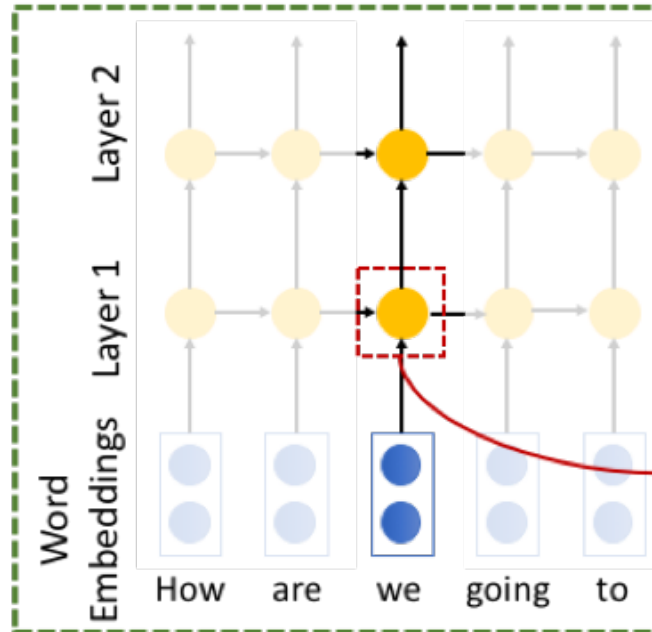
- Setup
 - f : an RNN encoder-decoder MT model
 - x and y are source and target sentences (lists of words)
 - g : a non-linear classifier (MLP with one hidden layer)
 - z : linguistic properties of words in x or y
- Morphology:
 - A challenge for machine translation, previously solved with feature-rich approaches.
 - Do neural networks acquire morphological knowledge?
- Experiment
 - Take $f^l(x)$, an RNN hidden state at layer l
 - Predict z , a morphological tag (*verb-past-singular-feminine*, *noun-plural*, etc.)
 - Compare accuracy at different layers l

Example: Machine Translation

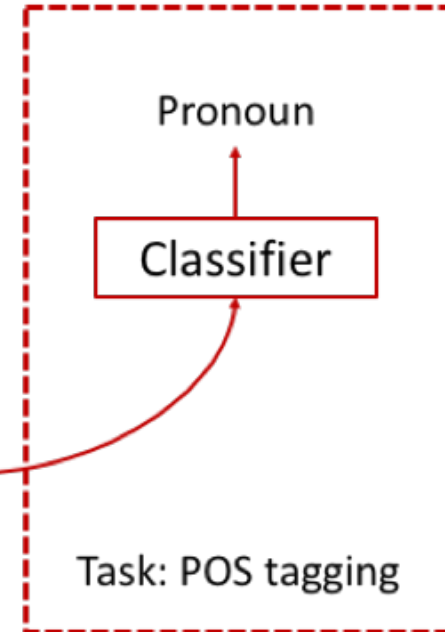
1. Train a neural MT system



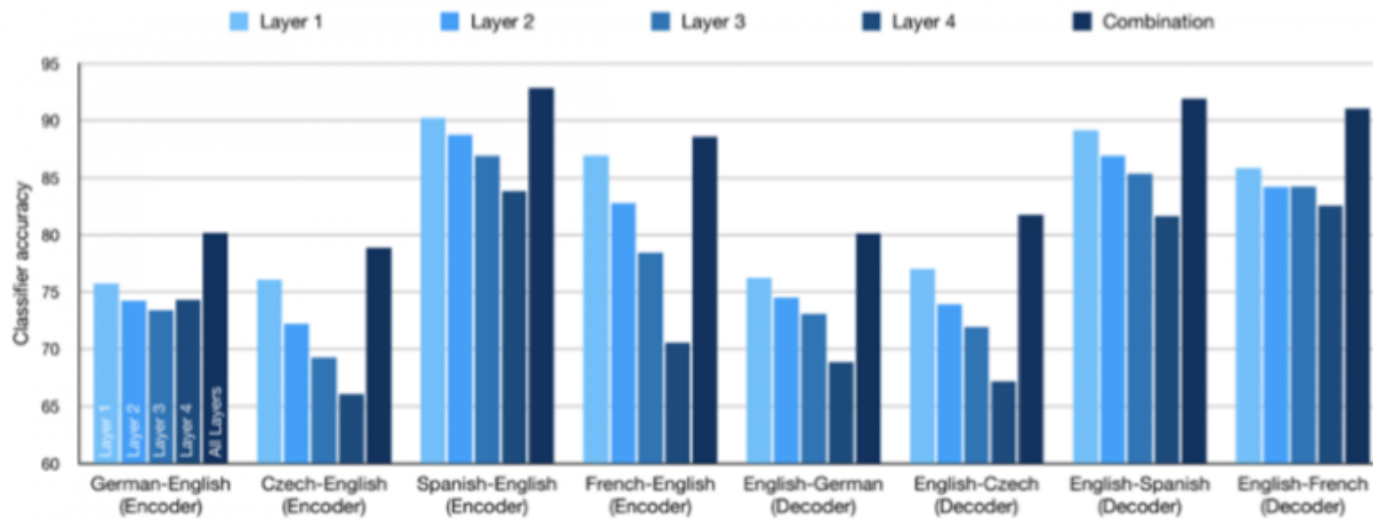
2. Generate feature representations using the trained model



3. Train classifier on an extrinsic task using generated features



Machine Translation: Morphology

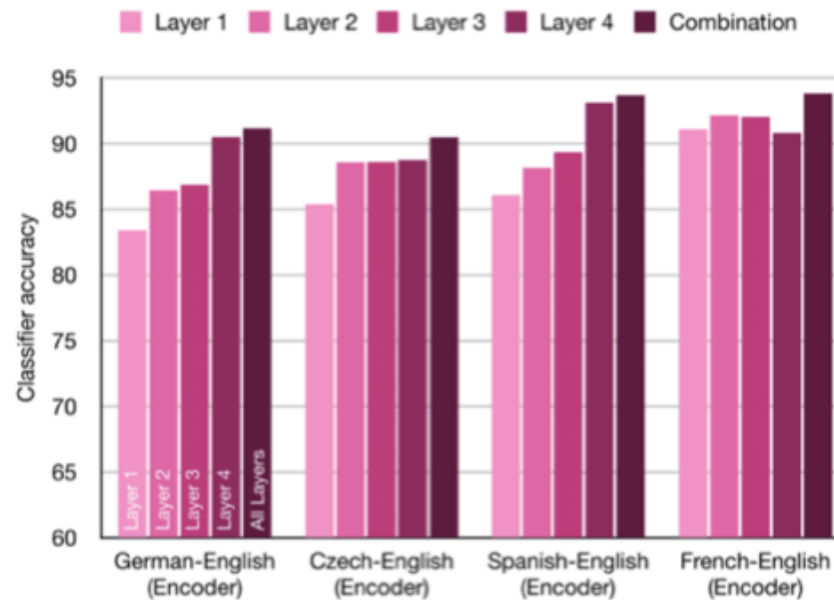


- Lower is better
- But deeper models translate better → what's going on in top layers?

Example: Machine Translation

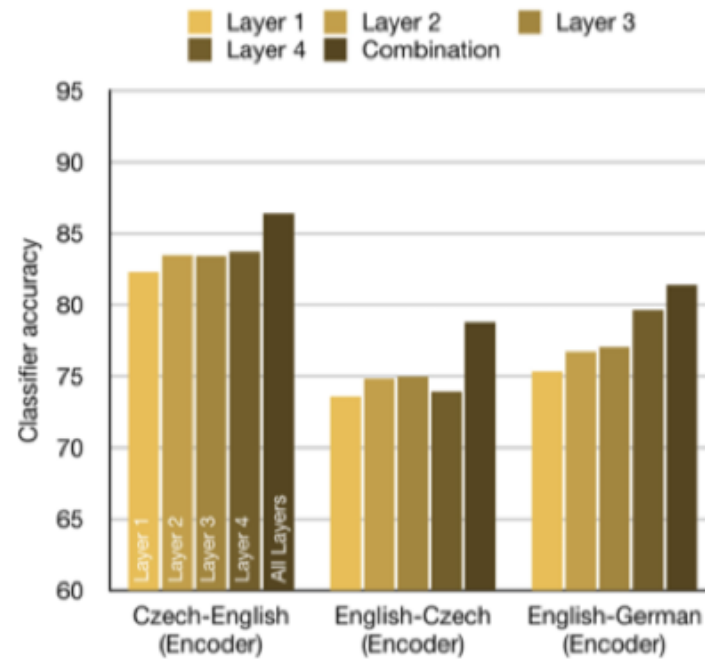
- Setup
 - f : an RNN encoder-decoder MT model
 - x and y are source and target sentences (lists of words)
 - g : a non-linear classifier (MLP with one hidden layer)
 - z : linguistic properties of words in x or y
- Syntax:
 - A challenge for machine translation, previously solved with hierarchical approaches.
 - Do neural networks acquire syntactic knowledge?
- Experiment
 - Take $[f(x_i) ; f(x_j)]$, RNN hidden states of words x_i and x_j , at layer l
 - Predict z , a dependency label (*subject*, *object*, etc.)
 - Compare accuracy at different layers l

Machine Translation: Syntactic Relations



- Higher is better

Machine Translation: Semantic Relations



- Higher is better

Hierarchies

Language
Hierarchy

Semantics

Discourse

Propositions

Roles

Syntax

Trees

Phrases

Relations

Morpho-Syntax

Parts-of-speech

Morphology

Lexicon

Hierarchies

Speech
Hierarchy

Words

Syllables

Phonemes

Complex

Simple

Articulatory features

Place

Manner

⋮

Language
Hierarchy

Semantics

Discourse

Propositions

Roles

Syntax

Trees

Phrases

Relations

Morpho-Syntax

Parts-of-speech

Morphology

Lexicon

Vision
Hierarchy

Scenes



Objects



Object parts



Motifs



Edges



Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- Suppose we get an accuracy, what should we compare it to?
 - Many studies focus on relative performance (say, comparing different layers)
 - But it may be desirable to compare to external numbers
 - **Baselines**: Often, compare to using static word embeddings ([Belinkov et al. 2017](#)) or random features ([Zhang and Bowman 2018](#))
 - This tells us that a representation is non-trivial
 - **Skylines**: Sometimes, report the state-of-the-art on the task, or train a full-fledged model
 - This can tell us how much is missing from the representation

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- Suppose we get an accuracy, what should we compare it to?
 - [Hewitt and Liang \(2019\)](#) define control tasks: tasks that only g can learn, not f
 - Specifically, assign a random label to each word type
 - A “good” probe should be selective: high linguistic task accuracy, low control task accuracy
 - Example
 - Linear vs. MLP
 - Accuracy vs. selectivity

Part-of-speech Tagging				
Model	Linear		MLP-1	
	Accuracy	Selectivity	Accuracy	Selectivity
Proj0	96.3	20.6	97.1	1.6
ELMo1	97.2	26.0	97.3	4.5
ELMo2	96.6	31.4	97.0	8.8

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- What is g ? What is the relation between the probe g and the model f ?
 - Common wisdom: use a linear classifier to focus on the representation and not the probe
 - Anecdotal evidence: non-linear classifiers achieve better probing accuracy, but do not change the qualitative patterns ([Conneau et al. 2018](#), [Belinkov 2018](#))

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- What is g ? What is the relation between the probe g and the model f ?
 - [Pimentel et al. \(2020\)](#) argue that we should always choose the most complex probe g , since it will maximize the mutual information $I(h; z)$, where $f(x)=h$

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- What is g ? What is the relation between the probe g and the model f ?
 - [Pimentel et al. \(2020\)](#) argue that we should always choose the most complex probe g , since it will maximize the mutual information $I(h; z)$, where $f(x)=h$
 - They also show that $I(x; z) = I(h; z)$ (under mild assumptions)
 - Thus the representation $f(x)=h$ contains the same amount of information about z as x
 - Does this make the probing endeavor obsolete?

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- What is g ? What is the relation between the probe g and the model f ?
 - [Pimentel et al. \(2020\)](#) argue that we should always choose the most complex probe g , since it will maximize the mutual information $I(h; z)$, where $f(x)=h$
 - They also show that $I(x; z) = I(h; z)$ (under mild assumptions)
 - Thus the representation $f(x)=h$ contains the same amount of information about z as x
 - Does this make the probing endeavor obsolete?
 - Not necessarily:
 - We would still like to know how good a representation is in practice
 - We can still ask relative questions about ease of extraction of information

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- What is g ? What is the relation between the probe g and the model f ?
 - [Voita and Titov \(2020\)](#) measure both probe complexity and probe quality
 - Instead of measuring accuracy, estimate the minimum description length: how many bits are required to transmit z knowing $f(x)$, plus the cost of transmitting g
 - Variational code: incorporate cost of transmitting g
 - Online code: incrementally train g on more data



Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- What is g ? What is the relation between the probe g and the model f ?
 - [Voita and Titov \(2020\)](#) measure both probe complexity and probe quality
 - Instead of measuring accuracy, estimate the minimum description length: how many bits are required to transmit z knowing $f(x)$, plus the cost of transmitting g
 - Variational code: incorporate cost of transmitting g
 - Online code: incrementally train g on more data
 - Example
 - Layer 0 control: control accuracy is high (96.3) but at the expense of codelength (267)

	Accuracy	codelength
MLP-2, h=1000		
LAYER 0	93.7 / 96.3	163 / 267
LAYER 1	97.5 / 91.9	85 / 470
LAYER 2	97.3 / 89.4	103 / 612

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z

Probing Classifiers: Limitations

- Recall the setup:
 - Original model $f : x \rightarrow y$
 - Probing classifier $g : f(x) \rightarrow z$
 - g maximizes the mutual information between the representation $f(x)$ and property z
- Correlation vs. causation
 - The probing classifier setup only measures correlation between representation $f(x)$ and property z
 - It is not directly linked to the *behavior* of the model f on the task it was trained on, that is, predicting y
 - Some work found negative/lack of correlation between probe and task quality ([Vanmassenhove et al. 2017](#), [Cífka and Bojar 2018](#))
 - An alternative direction: intervene in the model representations to discover causal effects on prediction ([Giulianelli et al. 2018](#), [Bau et al. 2019](#), Vig et al., in progress)

Outline





- Structural analyses
- **Behavioral analyses**
- Interactive visualizations
- Other methods

Behavioral Analyses

- Usually, we measure the *average-case* performance of $f : x \rightarrow y$ on a test set $\{x,y\}$, drawn uniformly at random from some text corpus
- However, this can reward models for performing well on common phenomena, and hide the fact that they perform poorly on “the tail”
- Challenge sets, a.k.a test suites aim to cover diverse phenomena
 - Systematicity
 - Exhaustivity
 - Control over data
 - Inclusion of negative data
- Thus they facilitate *fine-grained* analysis of model performance
- And they have a long history in NLP evaluation (Lehmann et al. 1996, Cooper et al. 1996, ...)

Behavioral Analyses

- Key idea: Design experiments that allow us to make inferences about the model's representation based on the model's behavior.

Test Sample	Nearest Neighbor Training Samples		
			
Q: What color are the safety cones?	Q: What color are the cones?	Q: What color is the cone?	Q: What color are the cones?
GT Ans: green	GT Ans: orange	GT Ans: orange	GT Ans: orange
Predicted Ans: orange			
Generalization "Opportunities" in Visual Question Answering (VQA) Devi Parikh. GenDeep Workshop.			
Slide credit: Aishwarya Agrawal			

Brett knew what many waiters find



Brett knew that many waiters find.



Warstadt et al. (2020)

Behavioral Analyses

- **Benefits:**
 - Avoid “squinting at the data”. Objective criteria for what counts as “representing” a thing
 - Theory agnostic. No constraints on how you represent it (symbolic, neural, feature-engineered)
 - Interfaces well with linguistics and other fields. “We are all responsible for the same data”.
 - Practical--not whether the model represents a feature, but whether it uses it in the right way
- **Limitations**
 - What’s to blame, the model or the data? How do we know what generalizations are “fair”?
 - Only tells us *that* a model did/didn’t solve a task; few insights into *how* the model solved the task, or *why* it failed to
 - Hard to design tightly controlled stimuli, probing sets themselves can have artifacts
 - Risk of overfitting to the challenge sets

Behavioral Analyses

- See our [survey](#) for a categorization of many studies
 - Tasks
 - Especially machine translation and natural language inference
 - Linguistic phenomena
 - Morphology, syntax, lexical semantics, predicate-argument structure
 - Languages
 - Mostly focusing on English, some artificial languages, not much work on other languages
 - Scale
 - Ranging from hundreds to many thousands
 - Construction method
 - Either manual or programmatic

Tasks used as probing tasks

- Ideally, simple task interfaces which can support lots of model types
- Ideally, minimal need for training/finetuning on top of model being “probed”

Task	Example	Typical Use	Strengths	Limitations	E.g.
LM [Might add generation here, too]	The boy by the boats [is/*are] smiling.	Syntactic phenomena	No additional training on top of pretrained LM	Often uses ppl, so best for left-to-right language models. Harder to use for newer variants.	Linzen et al. (2016)
Acceptability	The boy by the boats [is/*are] smiling.	Syntactic and semantic phenomena	More flexible than LM across architectures; well studied in ling.	Usually requires additional training on top of LM.	Warstadt et al. (2020)
NLI	The boy is smiling. -> The boy [is/*is not] happy.	Semantics/pragmatics/world knowledge	Flexible, easy to "recast" many tasks to NLI; long history	Often awkward sentences/confounds; low human agreement	White et al. (2017)
QA		Semantics/pragmatics/world knowledge	Can be more natural than NLI; incorporates more context	Requires custom system architecture (e.g. reading documents)	
MT	The repeated calls from his mother should have alerted us. Les appels rep´ et´ es de sa m´ ere devraient nous avoir alertes.	Multilingual morpho-/lexico-/syntax	Only way of specifically probing cross-lingual systems	Often relies on manual eval (though recent approaches use probabilities similar to in LM tasks)	Isabelle et al. (2017)

Experimental Designs

- Tightly Controlled
 - Minimal Pairs/Counterfactuals
 - Pros: Few confounds, more easy to attribute difference to the phenomena itself
 - Cons: Can be hard to generate; may not exist in a way that is natural
 - Good for phonemes that manifest neatly in the grammar (syntactic agreement, studying gender bias), but less so for complex phenomena (common sense/world knowledge)

Gender Bias: [Rudinger et al. \(2018\)](#)

(1a) **The paramedic** performed CPR on **the passenger** even though **she/he/they** knew it was too late.

(2a) **The paramedic** performed CPR on **the passenger** even though **she/he/they** was/were already dead.

Subj.-Verb Agree.: [Marvin and Linzen \(2018\)](#)

- a. The farmer that the parents love swims.
- b. *The farmer that the parents love swim.

Veridicality: [White et al. \(2018\)](#)

Someone {knew, didn't know} that a particular thing happened.

Someone {was, wasn't} told that a particular thing happened.

Did that thing happen?

Experimental Designs

- Loosely Controlled
 - Average over sets with vs. without property of interest
 - Pros: Can consist of naturalistic data; can generate larger test sets
 - Cons: Contain artifacts, harder to attribute differences to target phenomena

FraCas: [Cooper et al. \(1996\)](#)

GLUE Diagnostic Set: [Wang et al. \(2019\)](#)

Diverse Natural Language Inference Corpus (DNC): [Poliak et al. \(2018\)](#)

Experimental Designs

- Adversarial Examples

- Design data sets (usually using minimal pairs or “perturbations”) that specifically emphasize a model’s weaknesses
- Pros: Provides practical analysis of model failures; can be used as training to improve model
- Cons: Sets age quickly and are model/data specific; “whack-a-mole” style progress

[Jia and Liang \(2017\)](#)

Adversarial NLI: [Nie et al. \(2019\)](#)

Construction Methods

- Sources of Data
 - Sentences drawn from existing corpora
 - Sentences drawn from existing benchmark sets/test suites
 - Templates
 - Manual Generation
- Example/Label Generation
 - Labels are given by-definition (e.g. if using templates or manual generation)
 - Automatically manipulate sentences and assume heuristic labels (+/- human filtering)
 - Purely automatic (e.g. adversarial)
 - Purely manual labeling (e.g. human generated examples)

Construction Methods

- Method: Entirely Manual
- Examples: [Build-It-Break-It](#), [Adversarial NLI](#)

Construction Methods

- Method: Semi-Automatic + Crowdsourcing
- Examples: [Poliak et al. \(2018\)](#), [Kim et al. \(2019\)](#)

Construction Methods

- Method: Templates
- Examples: [Ettinger et al. \(2018\)](#), [McCoy et al. \(2019\)](#)

Construction Methods

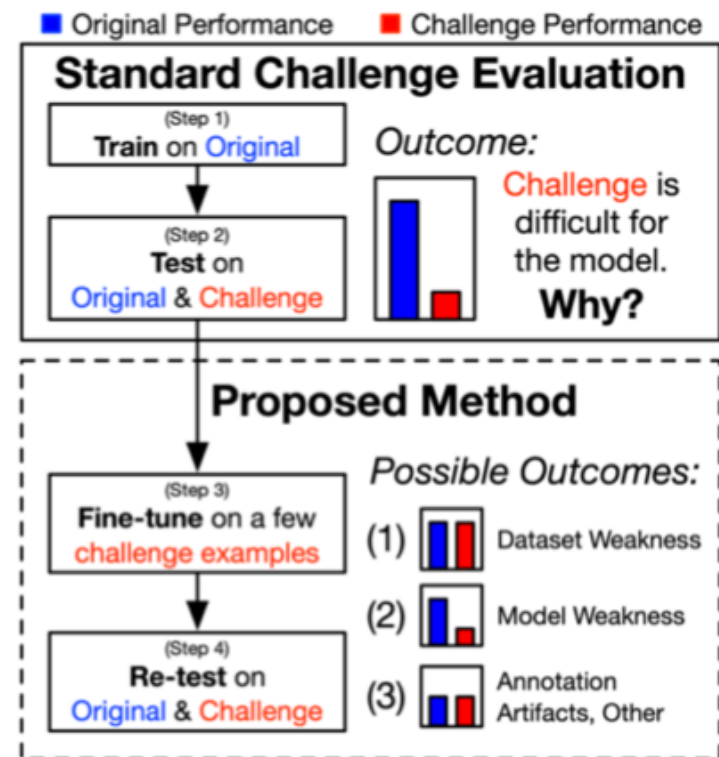
- Method: Entirely Automatic
- Examples: [Ebrahimi et al. \(2018\)](#), [Wallace et al. \(2019\)](#)

Challenge Sets: Limitations

- Availability
 - Limited coverage of tasks and languages
 - Need to expand beyond English and to more NLP tasks

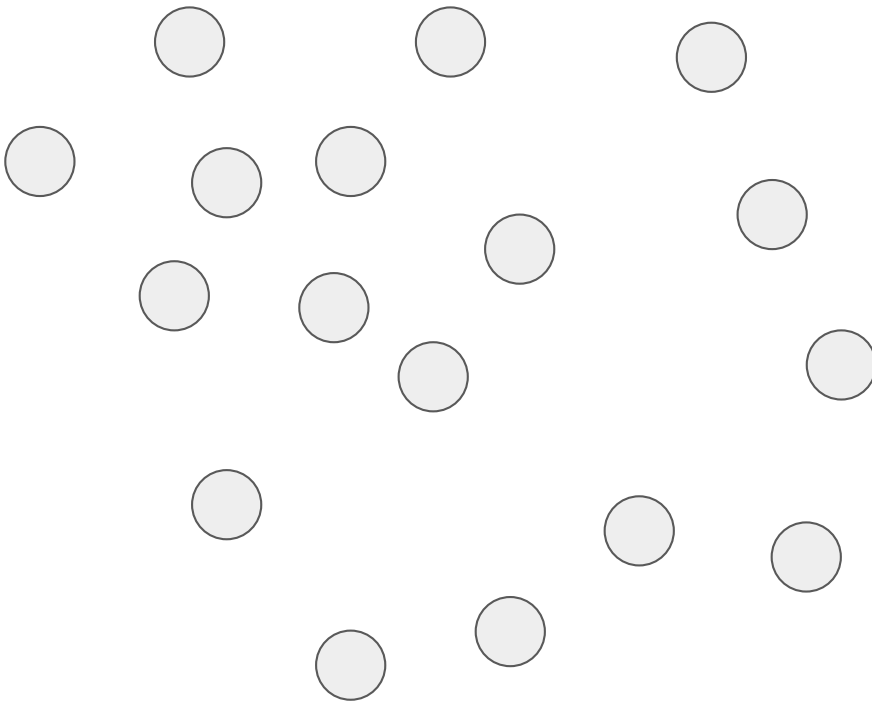
Challenge Sets: Limitations

- Availability
 - Limited coverage of tasks and languages
 - Need to expand beyond English and to more NLP tasks
- Methodology
 - What does failure on a challenge set tell us?
 - Who is to blame, the model or its training data?
 - [Lie et al. \(2019\)](#) fine-tune a model on a few challenge set examples and re-evaluate
 - [Rozen et al. \(2019\)](#) diversify both the training and test data
 - [Geiger et al. \(2019\)](#) propose method for determining whether a generalization task is “fair”

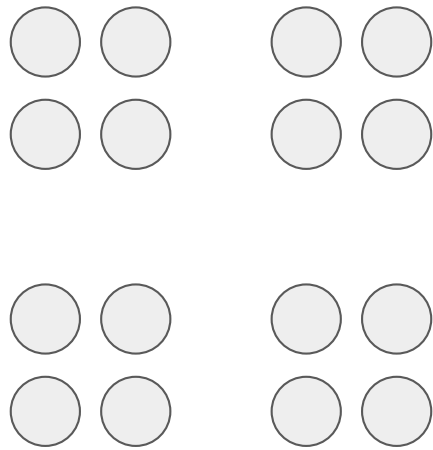


Outline

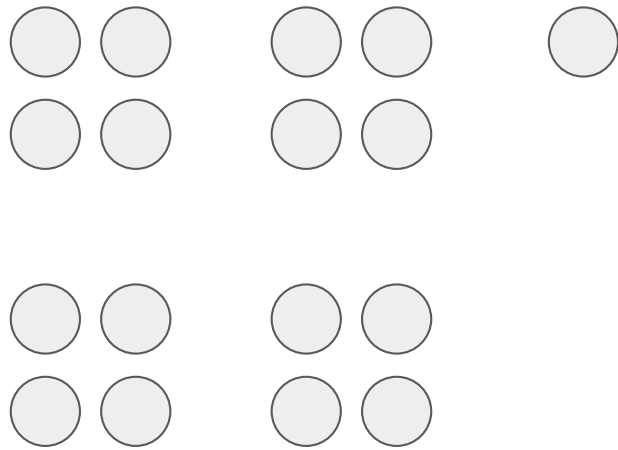
- Structural analyses
- Behavioral analyses
- **Interactive visualizations**
- Other methods



How many circles to you see?



Visualization can help you understand larger patterns



BUT... Visualization can lie. It was actually 17 🙄

Interaction and Visualization

1) Theory (Why, What)

2) Practice (How, Who)

- a) Attn=Explanation? Useful to look at either way
- b) Toolbox (Collaborators / Your own?)
- c) Practical Attn Vis Example: (1) agree on an API, (2) Code Server/Model, (3) Code Client/Frontend

3) Broader Perspective

- a) Reusable Vis: exBERT module (can we combine with mini vis? Install or so?)
- b) Automation through frameworks: Captum / AllenAI Interpret / LIT?
- c) Tighter integration with models? CSI

Why?

Interactive methods help:

- To generate hypotheses around model behavior or a dataset
- Reduce the exploration space when it is too large for brute-force methods
- Asking counterfactual “what if” question to the model and data

Interactive methods can:

- Enable the application of methods to real-world problems
- Lower the entry barrier through effective teaching
- Give visibility and feedback for new methods

Recent Tools:

[iNNvestigate](#)

[Captum](#)

[AllenNLP Interpret](#)

[exBert](#)

“A key element of the visualization approach is its ability to generate **trust** in the user. Unlike pure machine learning techniques, in a data visualization the user “sees” the data and information as a part of the analysis.

When the visualization is interactive, the user will be part of the loop and involved in driving the visualization. In such a context, the development of a **mental model** goes hand in hand with the visualization.“

	PA	PVA Exclusive
Overall Goal	Make Prediction	Support Explanation
Data Preprocessing	Clean and format data	Summarize and overview the training data
Feature Selection and Generation	Optimize prediction accuracy	Support reasoning and domain knowledge integration
Modeling	Optimize prediction accuracy	Support reasoning and domain knowledge integration
Result Exploration and Model Selection	Model quality analysis	Get insights; Select the proper model; Feedback for model updates
Validation	Test for overfitting	Get insights from other datasets

The tasks of a visual tool

Formulate Hypothesis

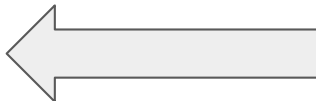
Groups of hidden states learn to capture linguistic properties



More accessible through “**playing**” with a model

Refine/Reject Hypothesis

The opening and closing of a parenthesis captured within a certain hidden state



Much **faster** with interactive tools

Compare models and datasets

Allow early generalization of insights



The design of the infrastructure of a VA tool can* be easily extensible to new models

User+Task Analysis does not just apply to vis!

Understand - Diagnose - Refine

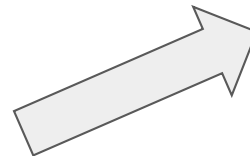
Towards better analysis of machine learning models: A visual analytics perspective.

[\[Liu et al. '17\]](#)



- 4.1 Interpretability & Explainability
- 4.2 Debugging & Improving Models
- 4.3 Comparing & Selecting Models
- 4.4 Teaching Deep Learning Concepts

WHY



- 5.1 Model Developers & Builders
- 5.2 Model Users
- 5.3 Non-experts

WHO

Architect - Trainer - End-User

LSTMVis: A Tool for Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks

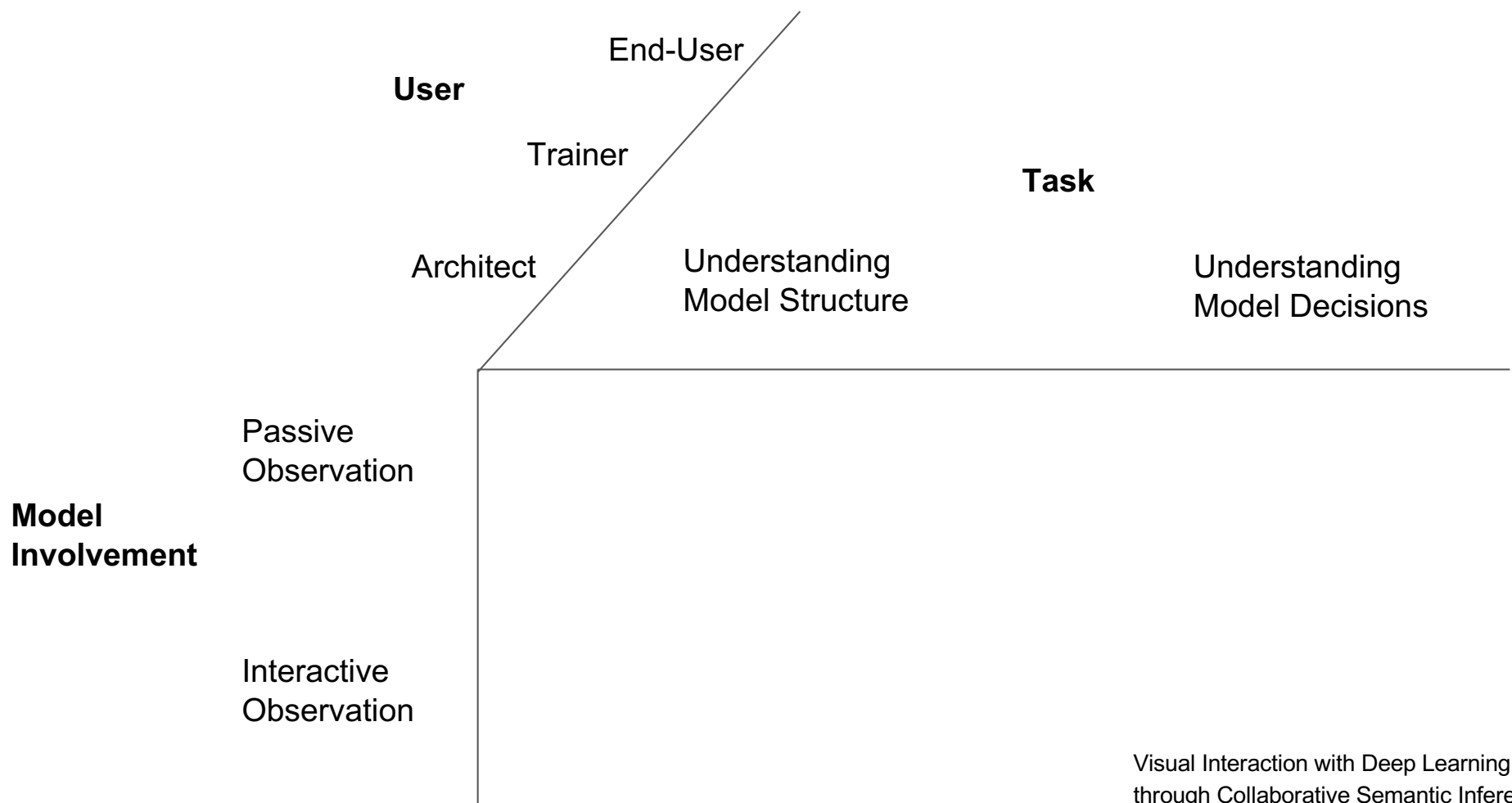
[\[Strobelt, Gehrmann, et al. '16\]](#)

- 6.1 Computational Graph & Network Architecture
- 6.2 Learned Model Parameters
- 6.3 Individual Computational Units
- 6.4 Neurons in High-dimensional Space
- 6.5 Aggregated Information

WHAT

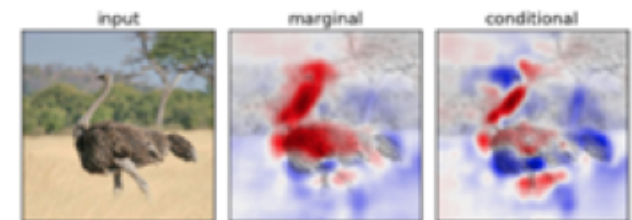
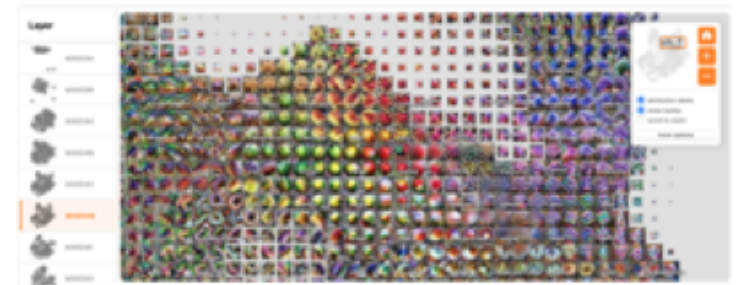
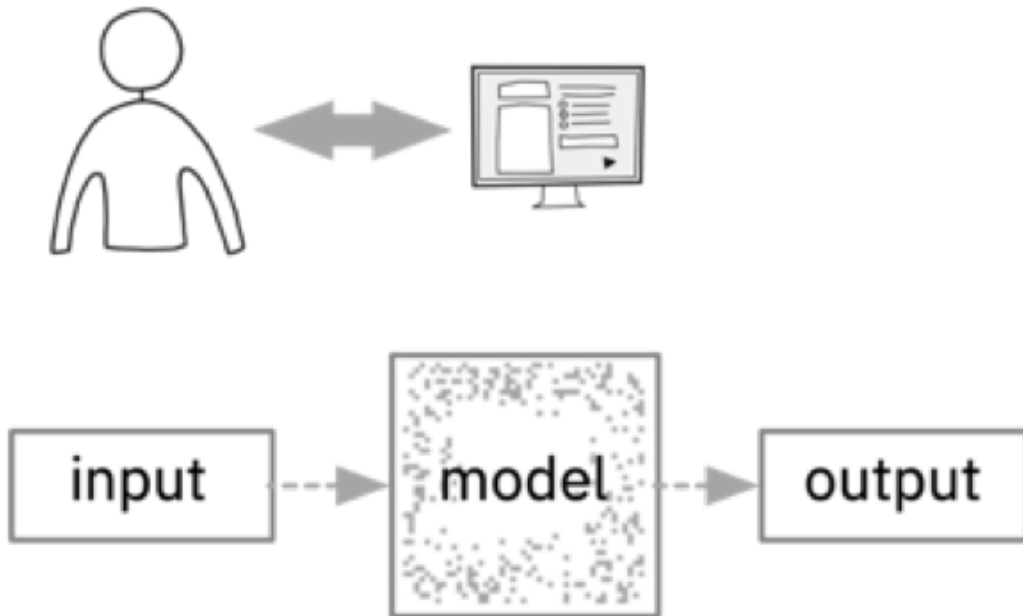
Visual analytics in deep learning: An interrogative survey for the next frontiers.

[\[Hohman et al. '18\]](#)

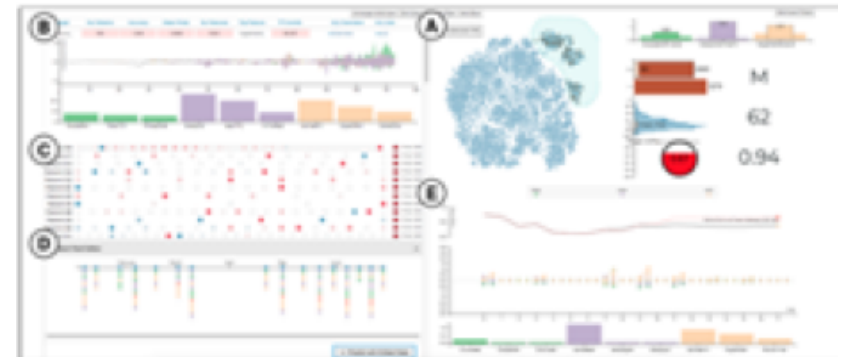
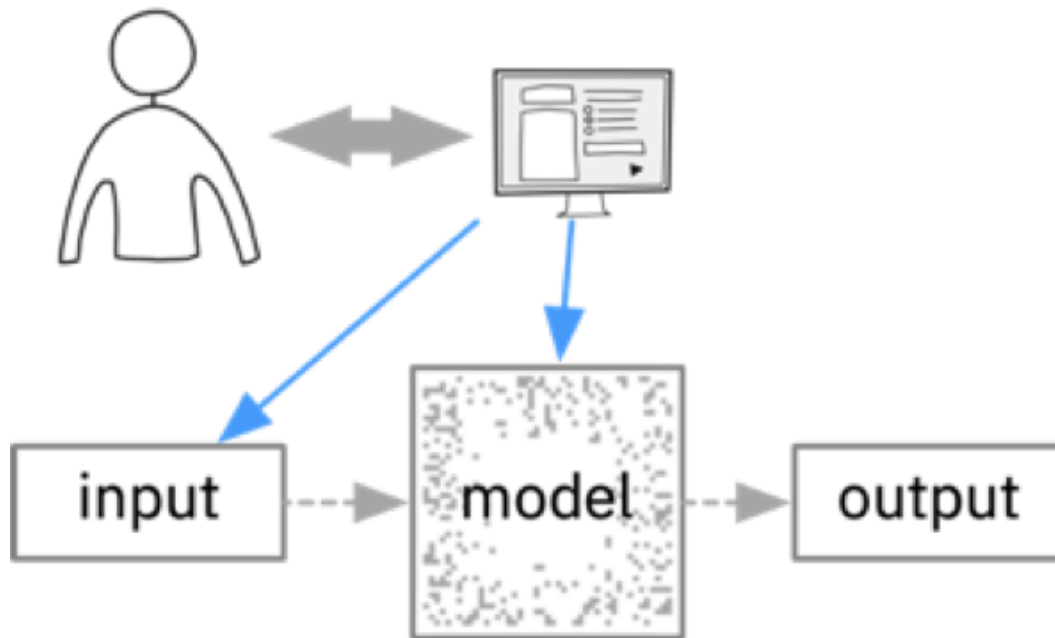


Visual Interaction with Deep Learning Models through Collaborative Semantic Inference.
[\[Gehrmann, Strobel, et al.'19\]](#)

Examples: Passive Observation



Examples: Interactive Observation



UX of Interaction (reading list)

Guidelines for Human-AI Interaction

[\[Amershi et al. '19\]](#)

Does the research process differ?

Vis/Interaction

Low Fidelity Prototypes

Goal-driven rapid iteration

Tons and tons of pilot studies



ML

Baseline Models

Dev-set driven hyperparameter exploration

Tons and tons of automatic evaluation

Note - Slide may contain cynical views on model development



The Unreasonable Effectiveness of Recurrent Neural Networks

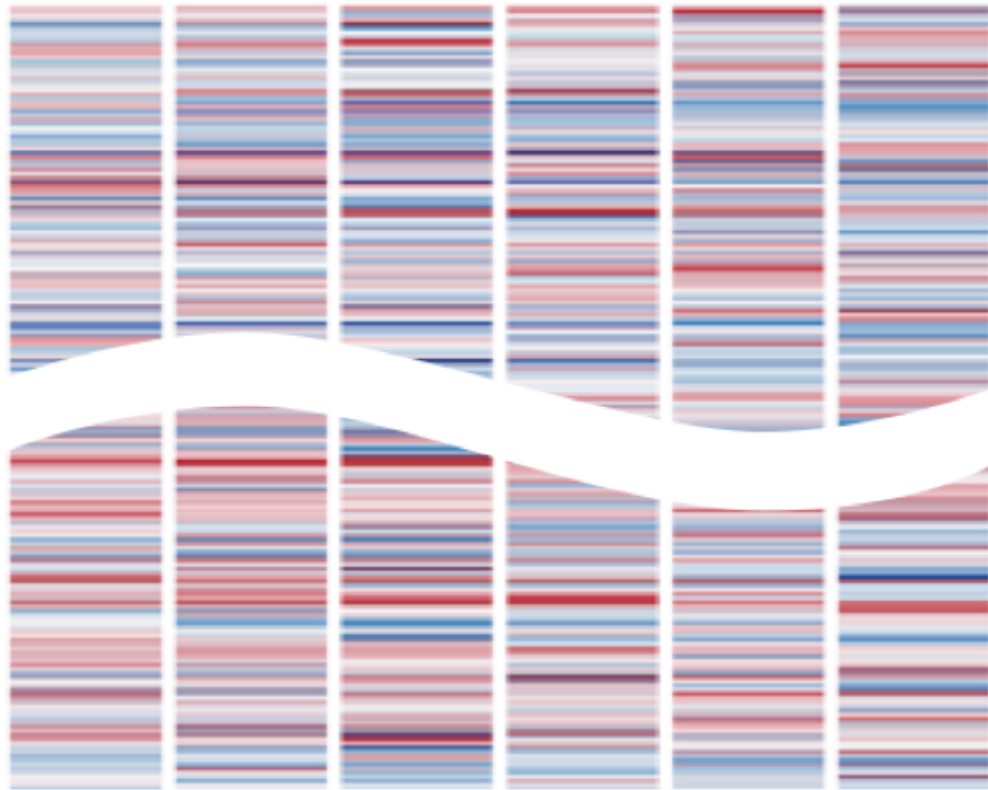
May 21, 2015

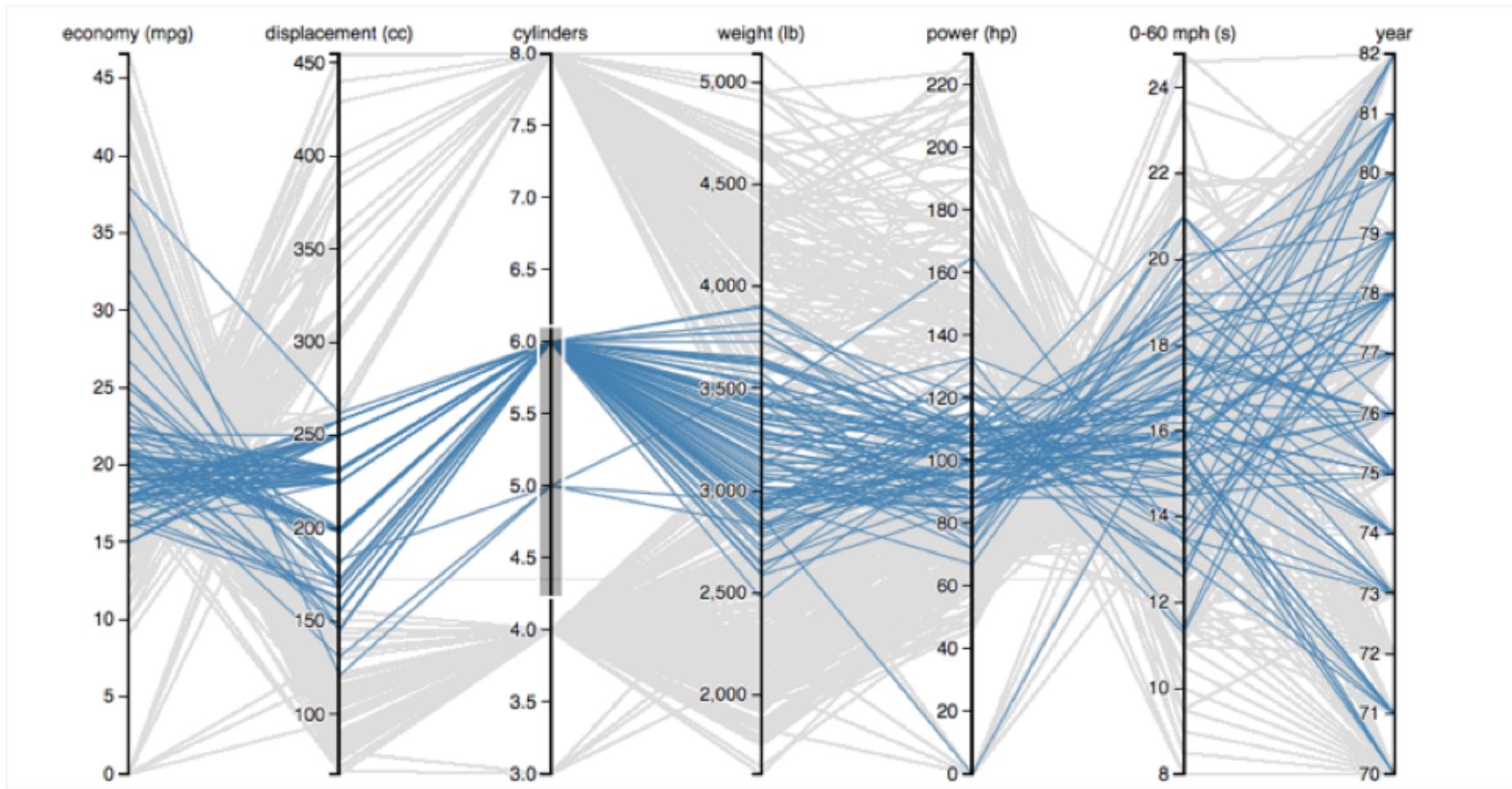
Cell that turns on inside quotes:

"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."

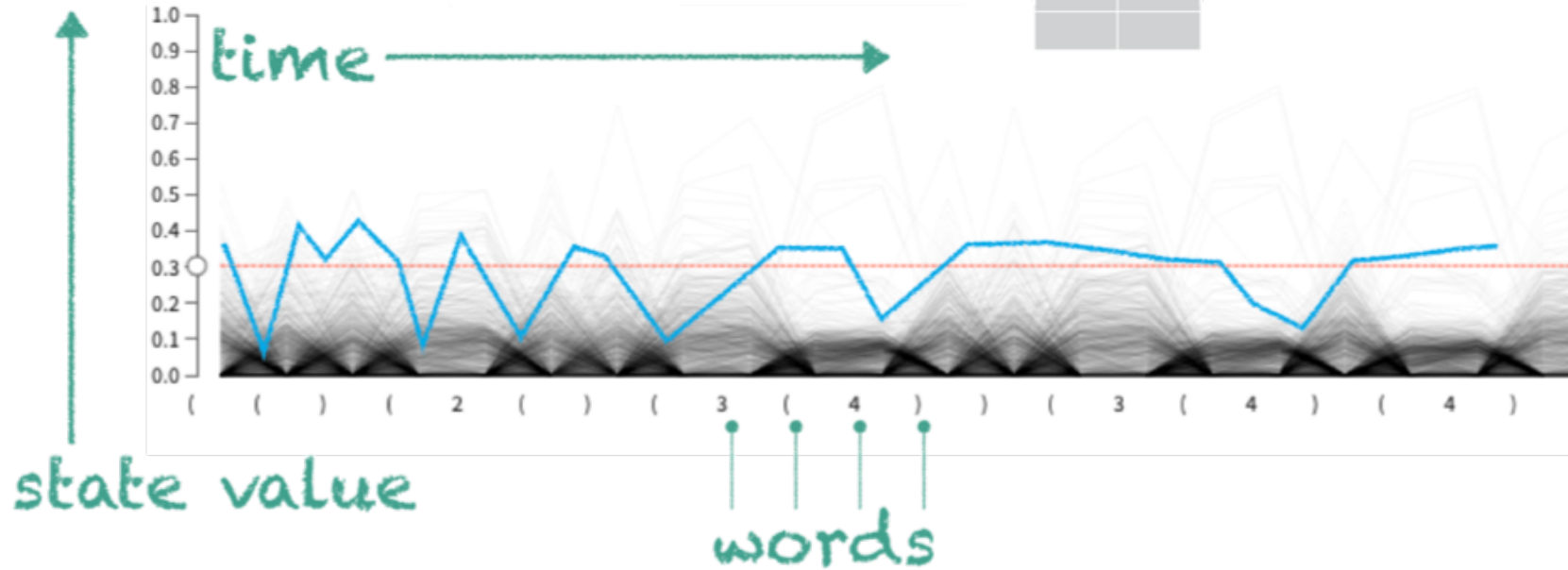
of the first aircraft is set

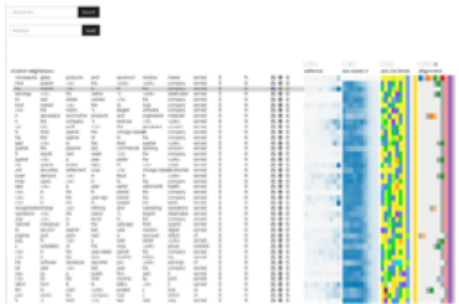




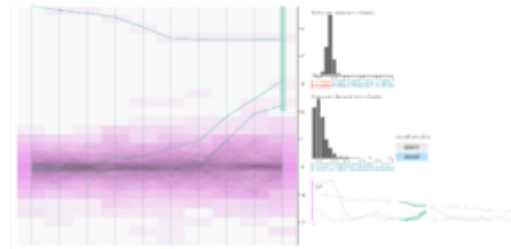
$h_{i...j}$

h_{t-1}	h_t
0.12	-0.4
0.01	0.07
0.1	0.3
...	

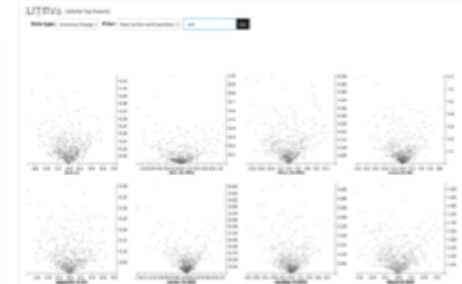




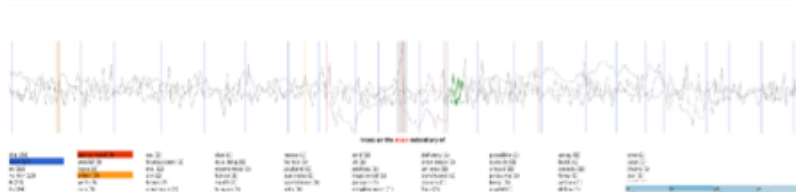
heatmap



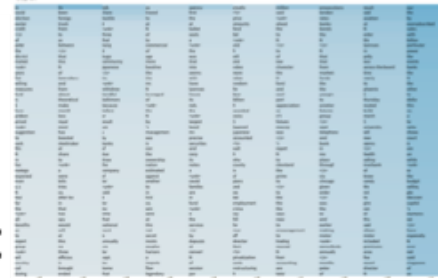
Ll plot



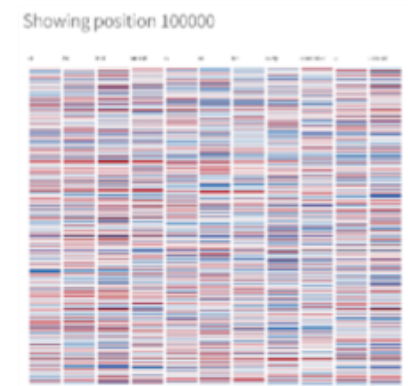
states comparison



linechart



gradient



states

Low Fidelity Prototypes

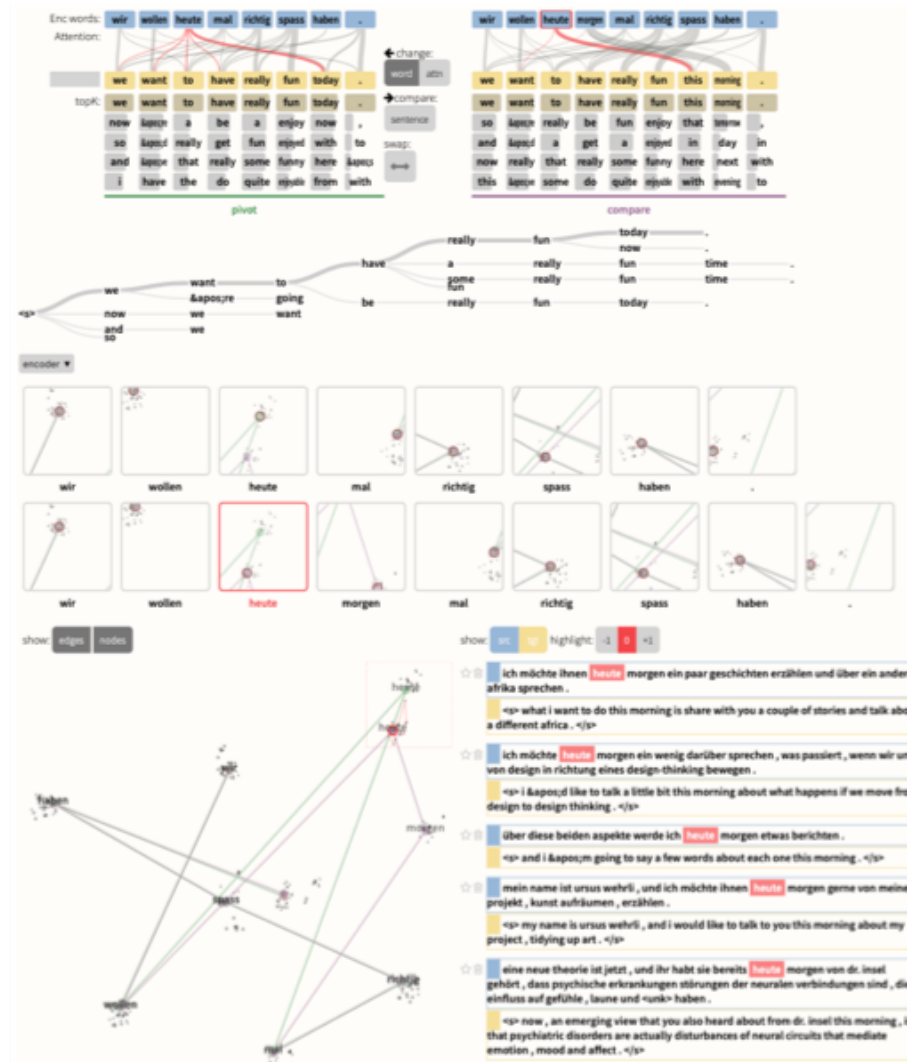
Goal-driven rapid iteration

Tons and tons of pilot studies

You have a cheap selection interface, now what?

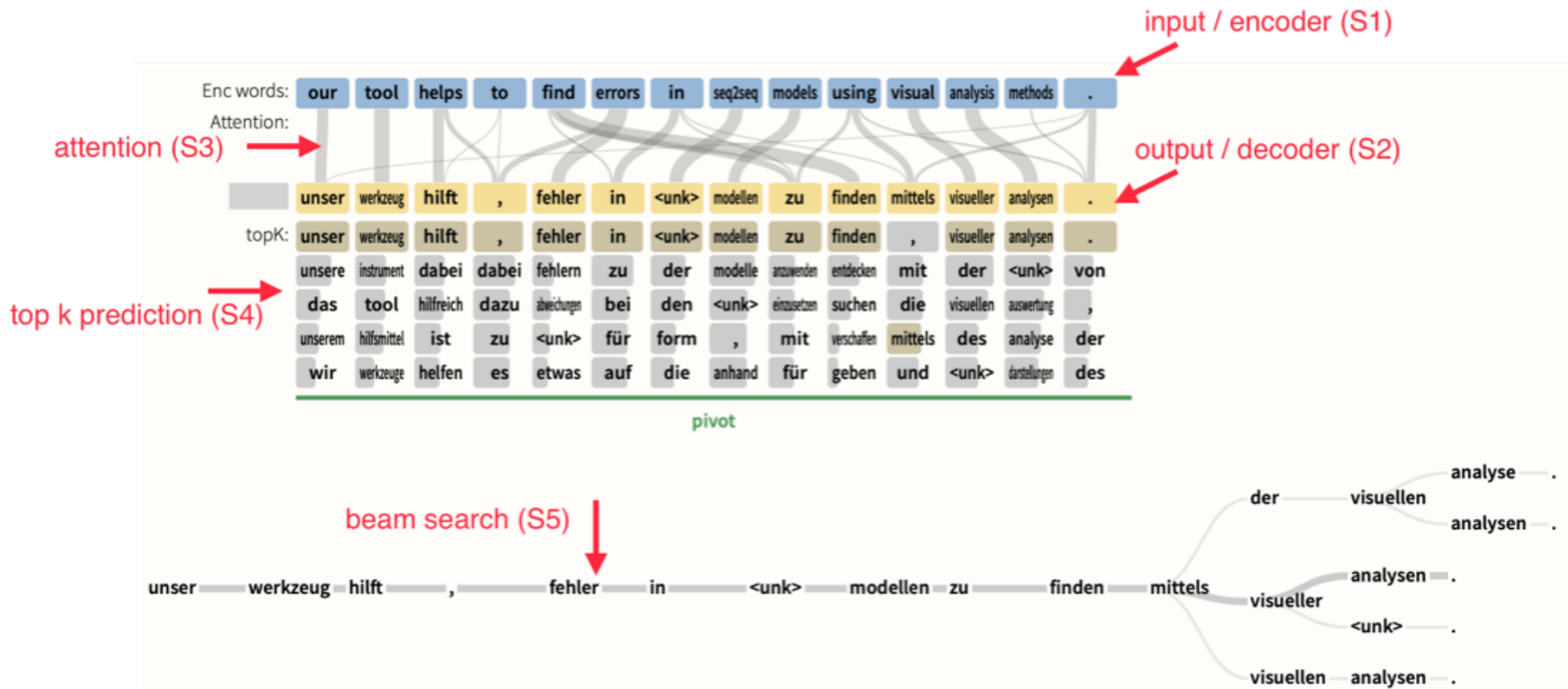
“cheap and universal”

Translation View (a)

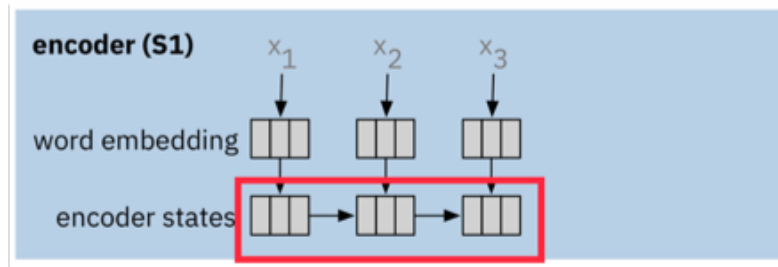


“expensive and deep”

Neighborhood View (b)



Connect Decisions to Previous Examples



lemma: training data = world view of model

pre-process: on trained model, record hidden state values for a large portion of the training data

inference: for new samples, find kNN of hidden states

[TODO: add overview of approaches here: Gradient-based, Influence Functions, challenge set reference for later, Statistics over entire corpus (kclark, iftenney), Train additional model]

Practical Attention Vis Example

Minimal Attention Vis

Select model:

Enter a sentence:

Results

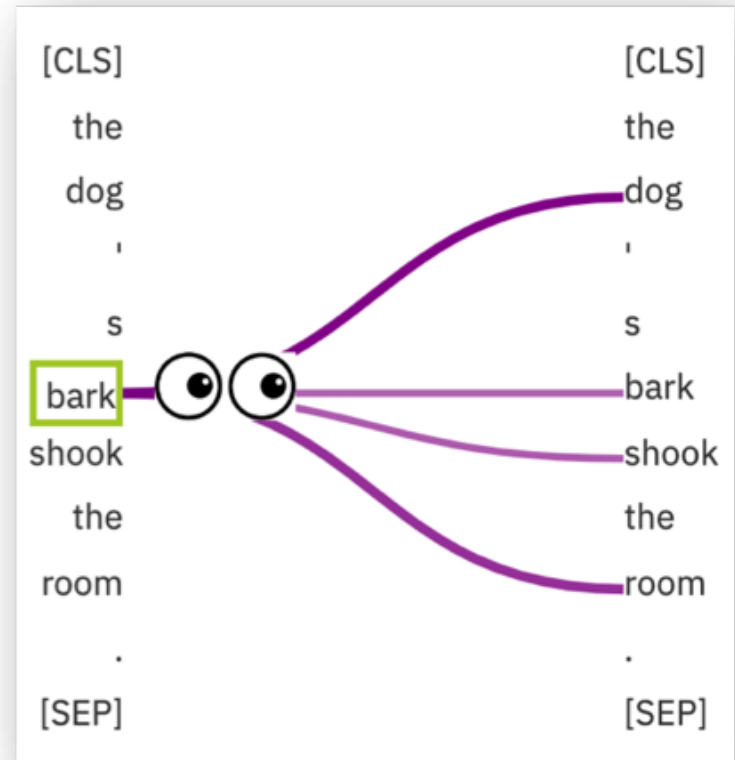
Layers & Heads

Challenges compared to s2s attention

Filtering: We now have 100+ heads

Aggregation: How do we show multiple?

Key/Value/Query: What do we do with that?



The 1-day JS Prototype

checkout github: <http://bit.ly/SIDN-AttnVis>

```
git clone https://github.com/SIDN-IAP/attnvis.git  
cd attnvis
```

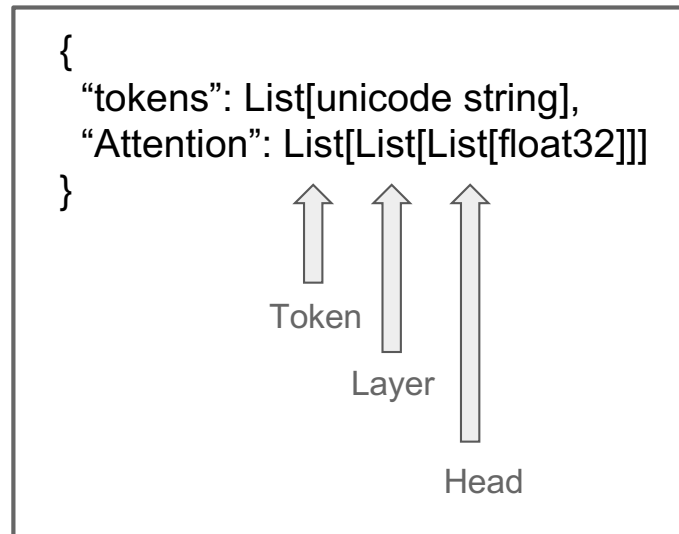
install dependencies:

```
conda env create -f environment.yml
```

get server to start without errors

```
conda activate attnvis  
python server.py
```

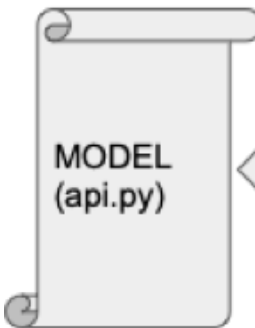
Agree on an API between backend and visualization



Note: this API does not support batching!

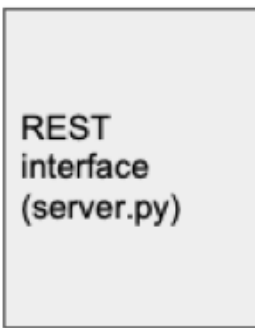
Python

Javascript / HTML / CSS



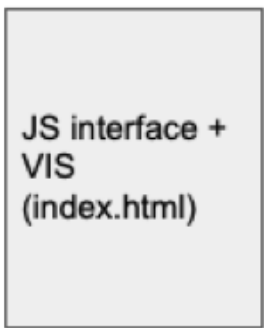
MODEL
(api.py)

huggingface
pytorch



REST
interface
(server.py)

flask



JS interface +
VIS
(index.html)

html/css/js
d3.js

Minimal Attention Vis

Select model: gpt-2

Enter a sentence: Life is what happens when you're busy making other plans.

Results

Life is what happens when you're busy making other plans.

Layers & Heads

0 1 2 3 4 5 6 7 8 9 10 11

The visualization shows a grid of attention weights for each layer (0-11) and head (0-11). The words 'what', 'happens', 'when', 'you're', 'busy', 'making' are highlighted in the results. The visualization shows that the model's attention is focused on these words across various layers and heads.



```
import torch
from transformers import AutoTokenizer, AutoModel

class AttentionGetter:
    '''Wrapper Class to store model object.'''
    def __init__(self, model_name: str):
        super().__init__()
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        self.model = AutoModel.from_pretrained(model_name, output_attentions=True).to(
            self.device
        )
        self.tokenizer = AutoTokenizer.from_pretrained(model_name)
```

api.py


```
def bert_analyze_text(self, text: str):
    """Works for BERT Style models"""
    # Tokenize input.
    toked = self.tokenizer.encode(text)
    # Build tensor and unsqueeze batch dimension.
    context = torch.tensor(toked).unsqueeze(0).long()
    # Extract attention.
    attn = self._grab_attn(context)
    # Build payload.
    return {
        "tokens": self.tokenizer.convert_ids_to_tokens(toked),
        "attention": attn,
    }
```

api.py

```

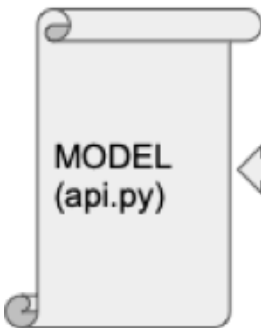
def _grab_attn(self, context):
    """
    function to get the attention for a model.
    First runs a forward pass and then extracts and formats attn.
    """
    output = self.model(context)
    # Grab the attention from the output tuple.
    # Format as Layer x Head x From/To
    attn = torch.cat([l for l in output[-1]], dim=0)
    format_attn = [
        [
            [[str(round(att * 100)) for att in head] for head in layer]
            for layer in tok
        ]
        for tok in attn.cpu().tolist()
    ]
    return format_attn

```

api.py

Python

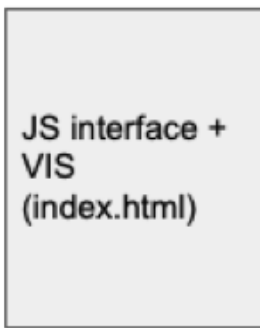
Javascript / HTML / CSS



MODEL
(api.py)



REST
interface
(server.py)



JS interface +
VIS
(index.html)

huggingface
pytorch

flask

html/css/js
d3.js

Minimal Attention Vis

Select model: gpt-2

Enter a sentence: Life is what happens when you're busy making other plans.

Results

Life is what happens when you're busy making other plans.

Layers & Heads

0 1 2 3 4 5 6 7 8 9 10 11

```
import json
import os

from flask import Flask as Flask,
from flask import request, redirect

from api import AttentionGetter

app = Flask(__name__)

# Set up cache for model wrappers.
loaded_models = {}

# redirect requests from root to index.html
@app.route('/')
def hello_world():
    return redirect('client/index.html')

if __name__ == '__main__':
    app.run()
```

server.py

```
@app.route('/api/attn', methods=['POST'])
def attn():
    sentence = request.json['sentence']
    model_name = request.json.get('model_name', 'distilbert-base-uncased')

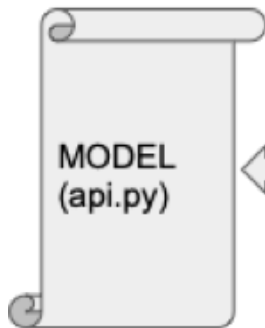
    # lazy loading.
    if model_name not in loaded_models:
        loaded_models[model_name] = AttentionGetter(model_name)
    model = loaded_models[model_name]

    # Call on the model to get attention
    results = model.bert_analyze_text(sentence)

    # return object with request (sentence, model_name) and results.
    return json.dumps({
        "request": {"sentence": sentence, 'model_name': model_name},
        "results": results
    })
```

server.py

Python



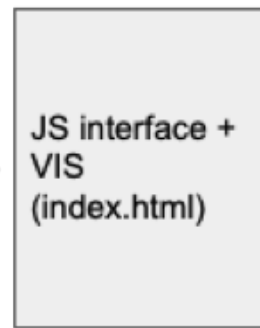
huggingface
pytorch



flask



Javascript / HTML / CSS



html/css/js
d3.js

Minimal Attention Vis

Select model: gpt-2

Enter a sentence: Life is what happens when you're busy making other plans.

Results

Life is what happens when you're busy making other plans.

Layers & Heads

0 1 2 3 4 5 6 7 8 9 10 11



```
<h3>Minimal Attention Vis</h3>
<div class="header">
  Select model: <select name="" id="model_select">
    <option value="gpt2"> GPT-2</option>
    <option value="distilbert-base-uncased"> DistilBert</option>
  </select>
  <br>
  Enter a sentence:
  <input type="text" id="inputText"
    value="I dropped my pen in the mashed potatoes.">
  <button id="sendButton"> send </button>
</div>
<hr>
<div style="padding-top: 5px;">
  <div style="font-weight: bold; padding-top: 10px;">Results</div>
  <div id="results" style="padding-top: 5px;">

  </div>
  <div style="font-weight: bold; padding-top: 10px;">Layers & Heads</div>
  <div id="layers" style="padding-top: 5px;">
  </div>
  <div id="heads" style="padding-top: 5px;">
  </div>
</div>
```

client/index.html

```

// select input field.
const myInput = d3.select("#inputText");
// act when content changes.
myInput.on('change', () => triggerServerRequest());
// also act on clicking the send button.
d3.select('#sendButton').on('click', triggerServerRequest);

function triggerServerRequest (){
  // get input content and bind to var.
  const input_sentence = myInput.property('value');
  const model_name = d3.select("#model_select").property('value');

  // send everything to the server
  // and return a promise
  const server_query = {
    method: "POST",
    body: JSON.stringify({
      sentence: input_sentence,
      model_name: model_name
    }),
    headers: {
      "Content-Type": "application/json"
    }
  };

  // if Promise is fulfilled (aka: server response is back) then...
  server_query.then(response => {
    currentModel = response.request.model_name;
    currentResults = response.results;

    // don't change selectedToken unless text is shorter
    selectedToken = Math.min(selectedToken,
      response.results.tokens.length - 1);

    // update layer buttons, heads visualization, text visualization
    updateLayerBtns(currentResults.attention.length);
    updateHeadsVis();
    updateTextVis();
  });
}

```

client/index.html


```

/**
 * update the layer buttons
 * @param no_btns -- number of buttons
 */
const updateLayerBtns = (no_btns) => {
  // create/update as many buttons as there are layers
  d3.select('#layers').selectAll('.btn').data(d3.range(no_btns))
    .join('div')
    .attr('class', 'btn')
    // most left/right buttons have round corners
    .classed('btn_l', d => d === 0)
    .classed('btn_r', d => d === (no_btns - 1))
    .text(d => d)
    .on('click', d => {
      // if clicked... set selected layer and update all VIS
      selectedLayer = d;
      updateLayerSelection();
      updateHeadsVis();
      updateTextVis();
    })

  updateLayerSelection();
};

```

- 1) *.selectAll* Select all *.btn* elements
[btn1, btn2, ...]
- 2) *.data* Set their data to the index value
[(btn1, 0), (btn2, 1), ...]
- 3) *.join* create/delete elements to match data
[(btn1, 0), (btn2, 1), ...]
- 4) *.classed* Conditionally set classes
- 5) *.text* Set their text to the index
- 6) *.on* Set their onClick handler

```
const updateLayerSelection = () => {  
  d3.select('#layers').selectAll('.btn')  
    .classed('selected', d => d === selectedLayer);  
}
```

- 1) *.selectAll* Select all *.btn* elements
[btn1, btn2, ...]
- 2) *.classed* Conditionally set classes

```

// defines gray scale for all heatmaps:
const colorScale = d3.scaleLinear().domain([0, 10, 100])
  .range(['#fff', '#aaa', '#4d4d4d'])

function updateHeadsVis() {
  // select heads vis root DOM node
  const headsDOM = d3.select('#heads');

  // in each heads Row add each token vis
  heads.selectAll('.attBox').data(d => d[selectedToken])
    .join('div')
    .attr('class', 'attBox')
    // highlight the token that has been selected
    .classed('selected', (d, i) => i === selectedToken)
    .style('background-color', d => colorScale(d))
}

```

Define a linear color scale variable

- 1) *.selectAll* Get all attention head elements
- 2) *.data* Filter attn values to those of the selected token and bind to head elements
- 3) *.join* Create/delete elements to match number of attention links
- 4) *.attr* Make sure all divs (even the just created one's) have the correct class
- 5) *.classed* highlight the selected token
- 6) *.style* Set the color to the color scale value

Minimal Attention Vis

Select model:

Enter a sentence:

Results

Layers & Heads

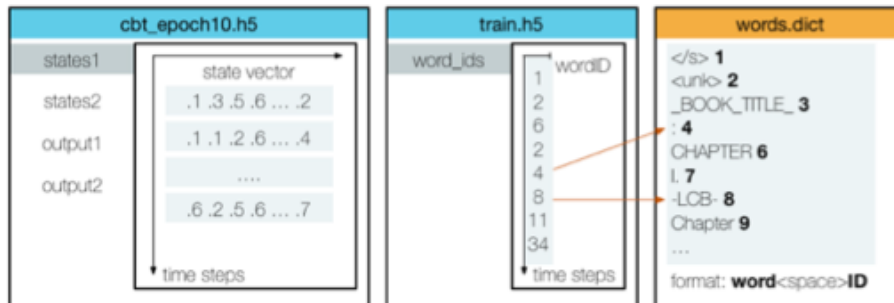
Call for Reproducibility and Public Adoption: open source with **documentation**

Adding Your Own Data

If you want to train your own data first, please read the [Training](#) document. If you have your own data at hand, adding it to LSTMVis is very easy. You only need three files:

- HDF5 file containing the state vectors for each time step (e.g. `cbt_epoch10.h5`)
- HDF5 file containing a word ID for each time step (e.g. `train.h5`)
- Dict file containing the mapping from word ID to word (e.g. `words.dict`)

A schematic representation of the data:



*If you don't have these files yet, but a space-separated `.txt` file of your training data instead, check out our [text conversion tool](#)

Config File

a simple example of an `lstm.yml` is:

```
name: children books # project name
description: children book texts from the Gutenberg project # little description

files: # assign files to reference name
states: cbt_epoch10.h5 # HDF5 files have to end with .h5 or .hdf5 !!!
word_ids: train.h5
words: words.dict # dict files have to end with .dict !!

word_sequence: # defines the word sequence
file: train # HDF5 file
path: word_ids # path to table in HDF5
dict_file: words # dictionary to map IDs from HDF5 to words

states: # section to define which states of your model you want to look at
file: states # HDF5 files containing the state for each position
types: [
  {type: state, layer: 1, path: states1}, # type={state, output}, layer=[1..x], path = HDF5 path
  {type: state, layer: 2, path: states2},
  {type: output, layer: 2, path: output2}
]
```

exBERT visualization component

[placeholder]

Broader Perspective / Current Opportunities

[TODO: add reading list items in categories]

Broader Perspective / Current Opportunities

Opportunity: Mixed-Initiative Guidance

Towards better analysis of machine learning models: A visual analytics perspective.

[\[Liu et al. '17\]](#)

Opportunity: Tighter integration of model + interface development

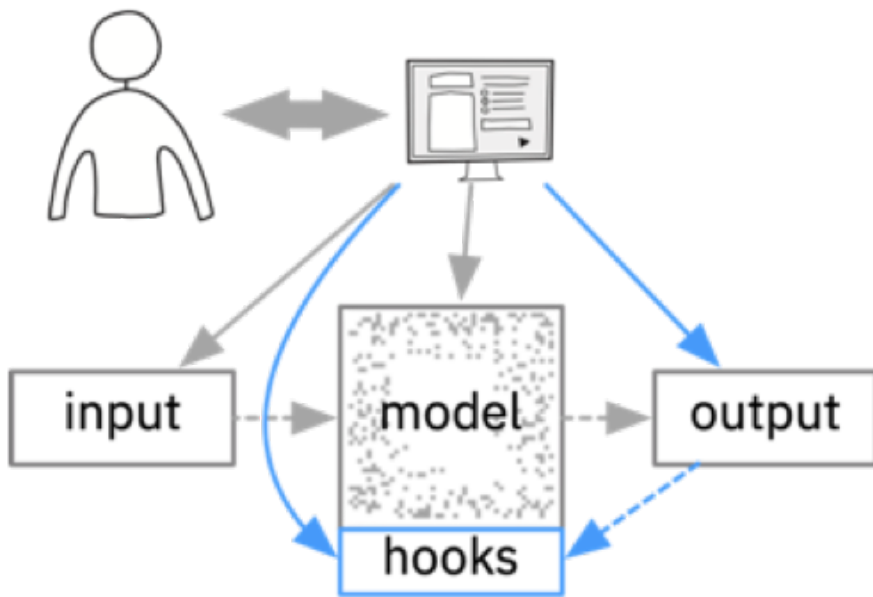
Placeholder

[Placeholder '20]

Test Alternative Decisions or WHAT IF Mode

Allow reasonable and interpretable modifications of your model input and internals during inference mode to help with understanding and debugging.

Interactive Collaboration



Interactive Visualization Questionnaire

What is the goal of the tool?

Pedagogical / Debugging / Debiasing / ...

Understanding model structure / model decisions / data / ...

How do you quantify an outcome?

Who is your user?

ML or NLP Expert/ Domain Expert / Student / ...

How much domain/ model knowledge do they have?

The answers will inform the following implementation questions:

Does the tool require interaction with the model? With the data?

Can you change the model structure or model decisions?

Outline

- Structural analyses
- Behavioral analyses
- Interactive visualizations
- **Other methods**

Other Topics

- Adversarial examples
 - Can point to model weaknesses
 - Challenges with text input (and output)
 - How to calculate gradients
 - How to measure similarity to real examples
 - Survey papers: [Belinkov and Glass 2019](#), [Wang et al. 2019](#), [Zhang et al. 2019](#)
- Generating explanations
 - Annotated explanations
 - Rationales: erasure-based, latent variables
- Formal languages as models of language
 - For example: can LSTMs learn context-free languages?
 - Long line of research starting in the 1980s

References and Resources

- Another upcoming tutorial (online):
 - Wallace, Gardner, and Singh, EMNLP 2020 tutorial on Interpreting Predictions of NLP Models

Conclusion