Understanding Large Language Models

Carsten Eickhoff, Michael Franke and Polina Tsvilodub

Session 07: Probing & Attribution

Main learning goals

1. Scene Setting
   - motivation, what-if exploration, global vs. local interpretability
2. Debugging Language Models
   - code, training issues, test issues, optimization/eval mismatch
3. Shapley Values
   - origin, intuition, limitations
4. Attention Visualization
   - attention recap, visualizations, limitations
5. Gradient Tracing
   - intuition, integrated gradients, CAM, Grad-CAM
6. Probing
   - probing classifiers, task types, control tasks, knowledge editing

Scene Setting

Visual Question Answering

Q. How symmetric are the white bricks on either side of the building?

Model answers: very
Ground truth: very

Thoughtfully constructed training data

200K images, 600K questions

Test accuracy of Kazemi and Elqursh (2017) model: **61%**

Right for the Wrong Reason

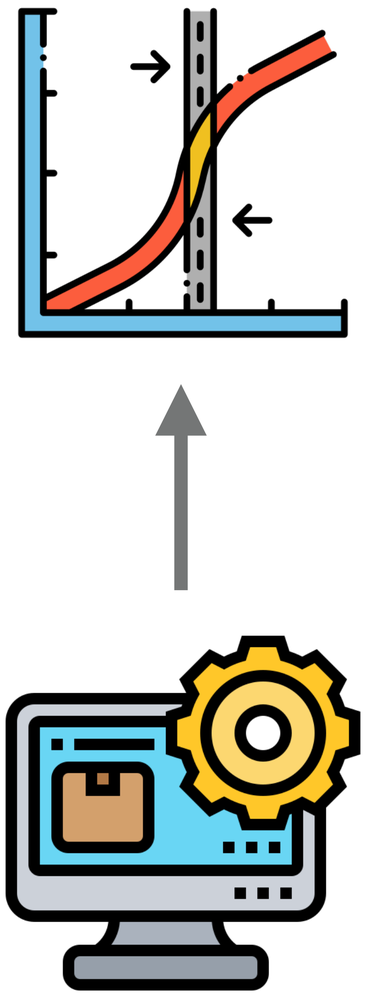
Q: “how asymmetric are the white bricks on either side of the building”
A: *very*

Q: “how soon are the bricks fading on either side of the building”
A: *very*

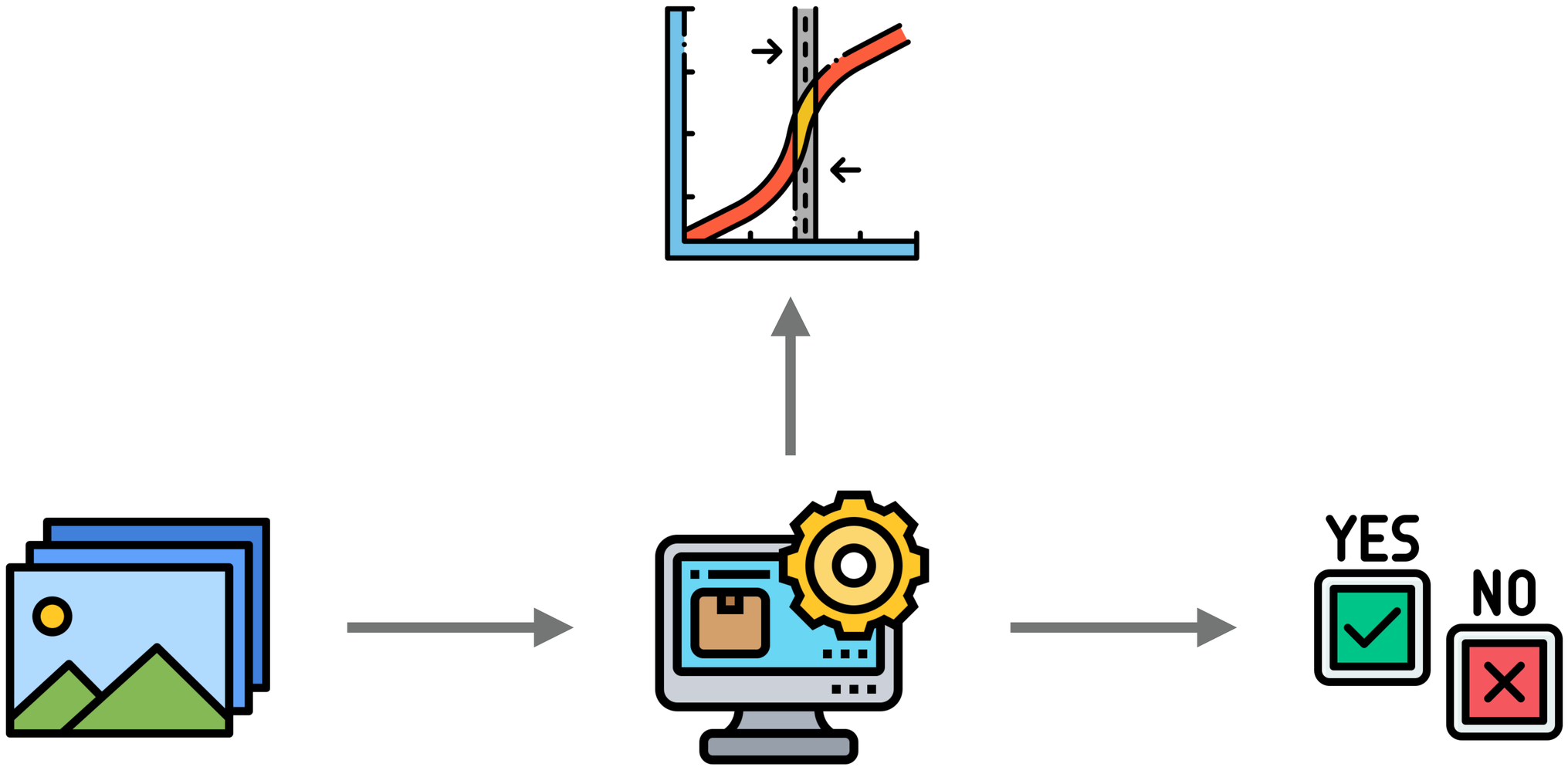
Q: “how fast are the bricks speaking on either side of the building”
A: *very*

Paper: Did the model understand the question? ACL 2018

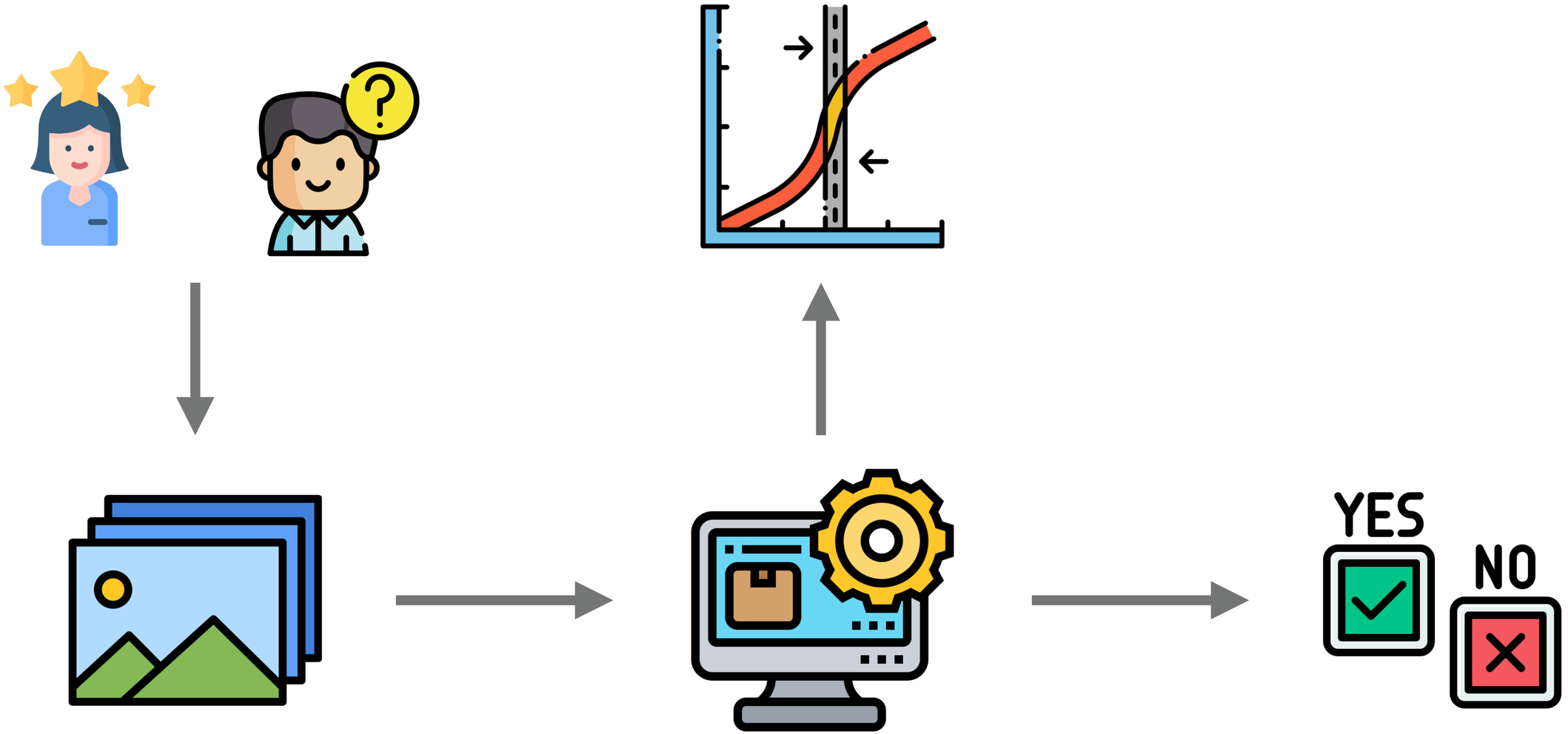
Global Explanations



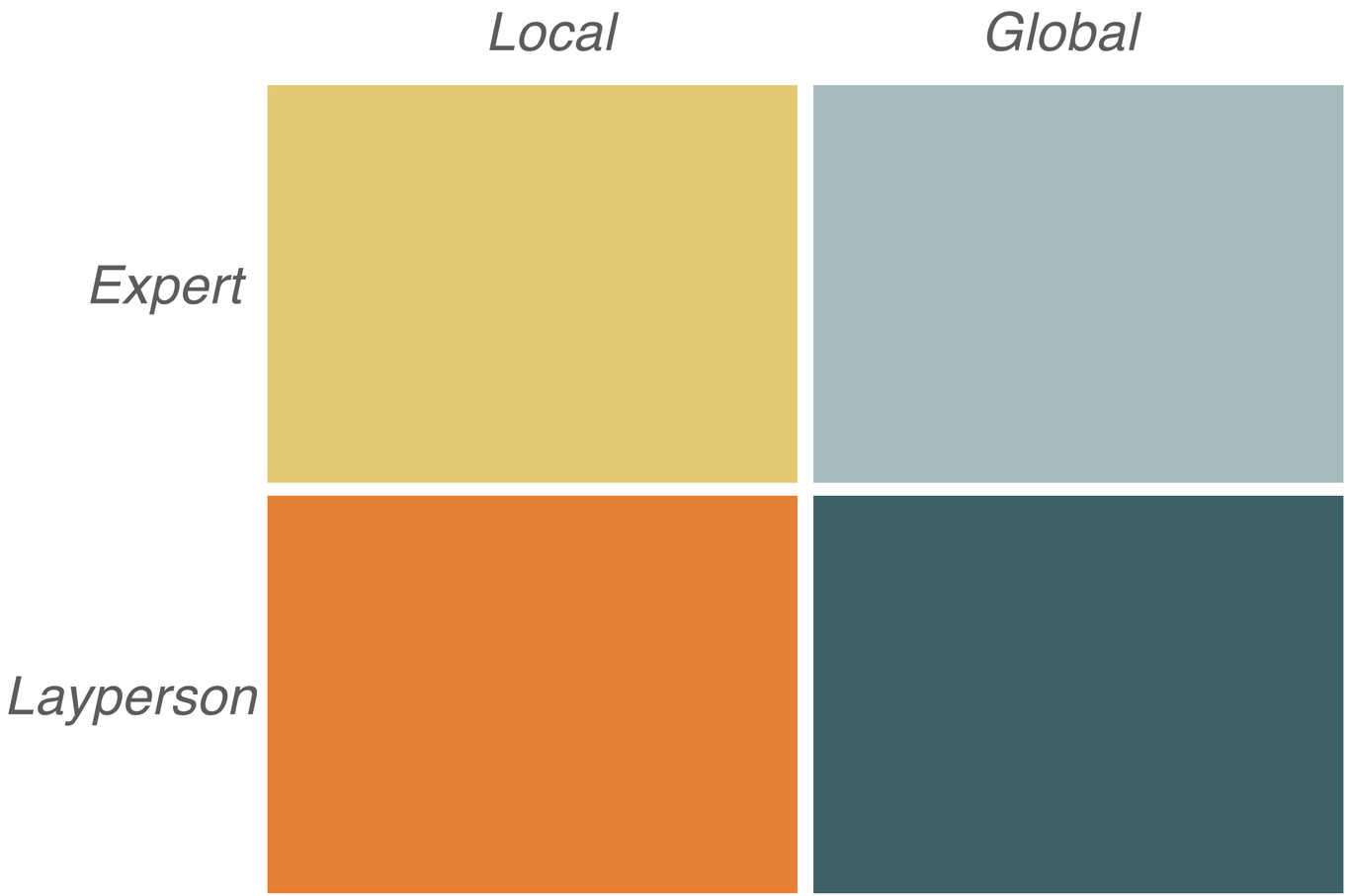
Local Explanations



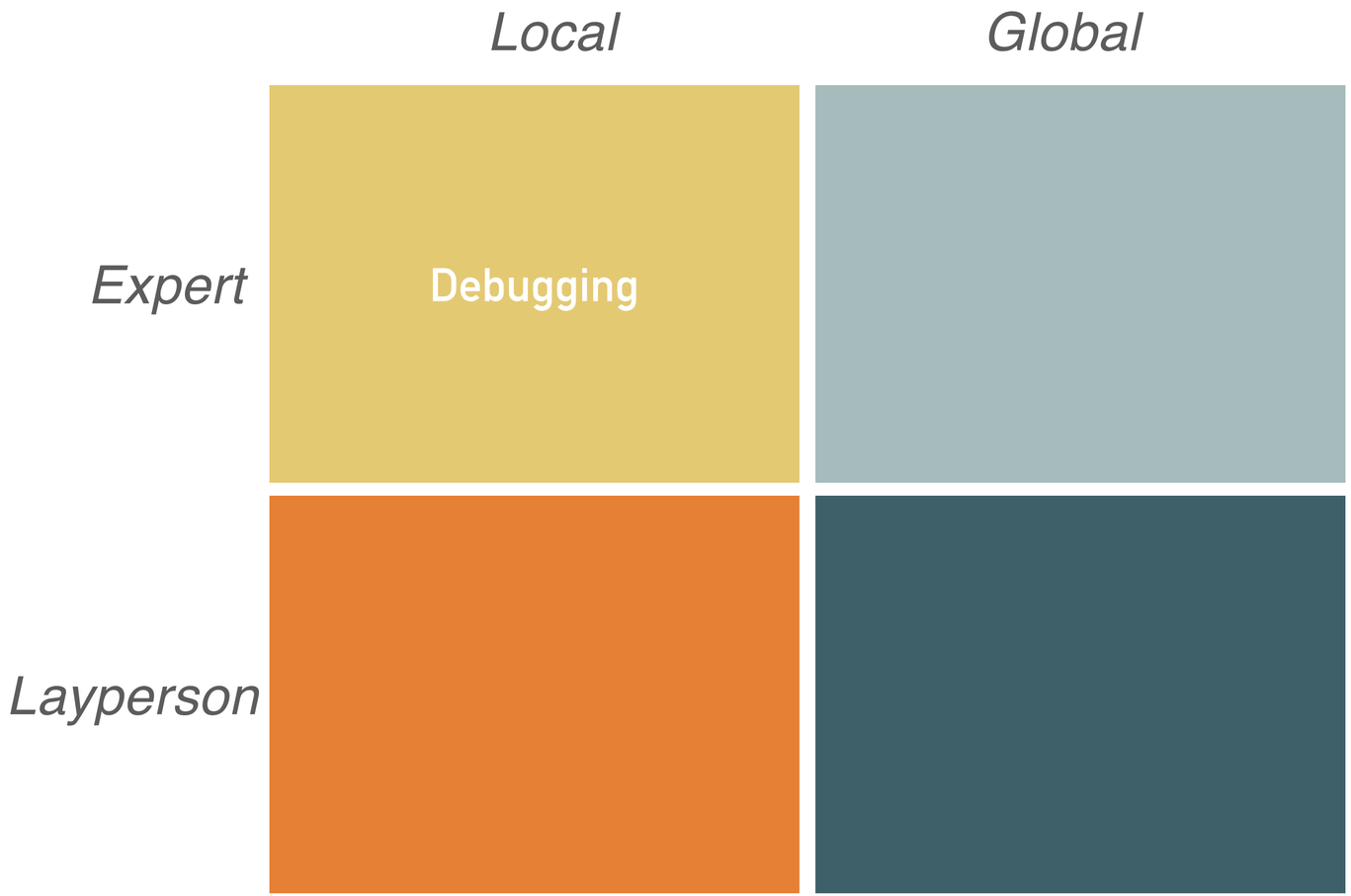
Suitable Explanations



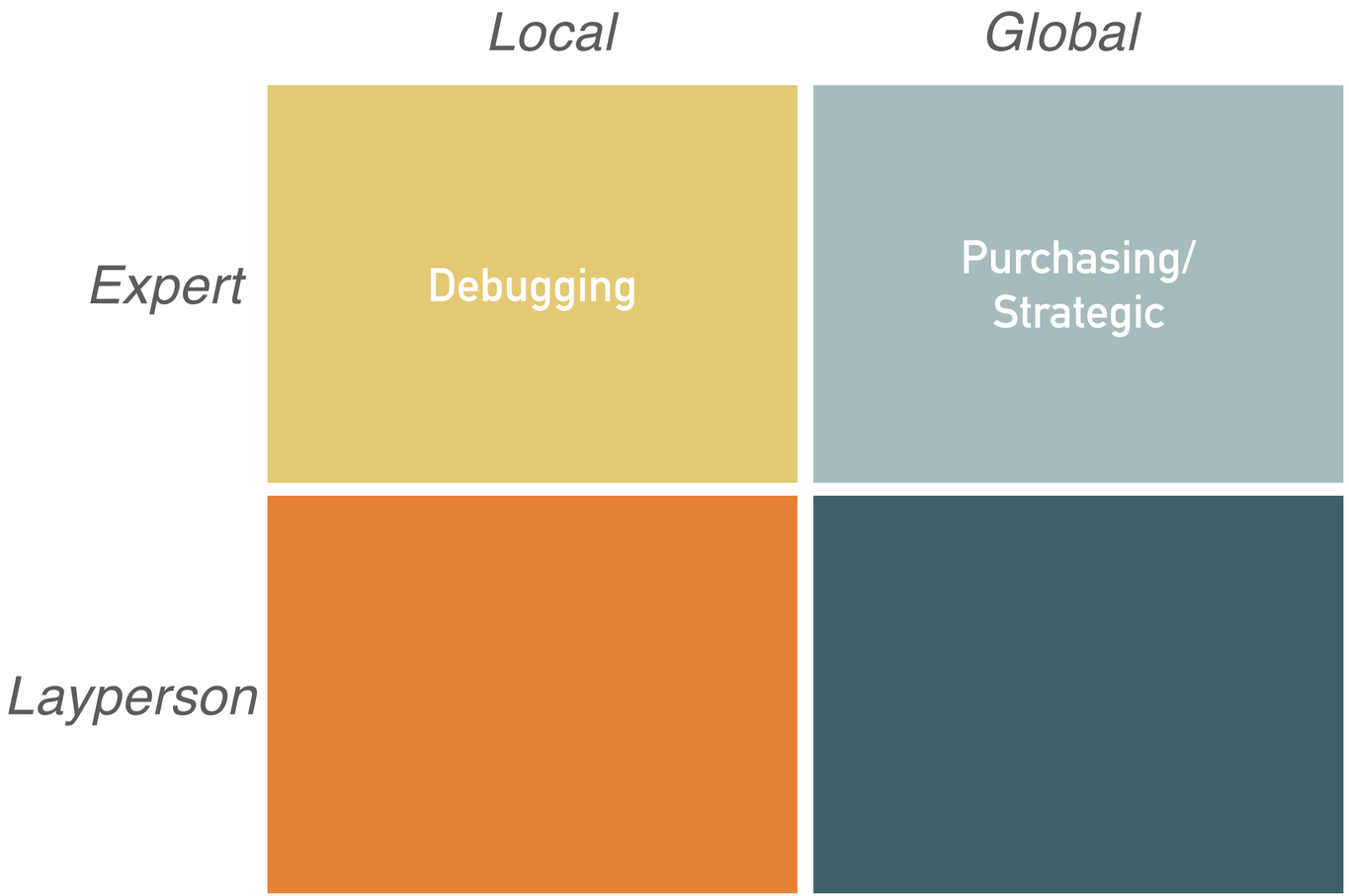
Tailored Explanations



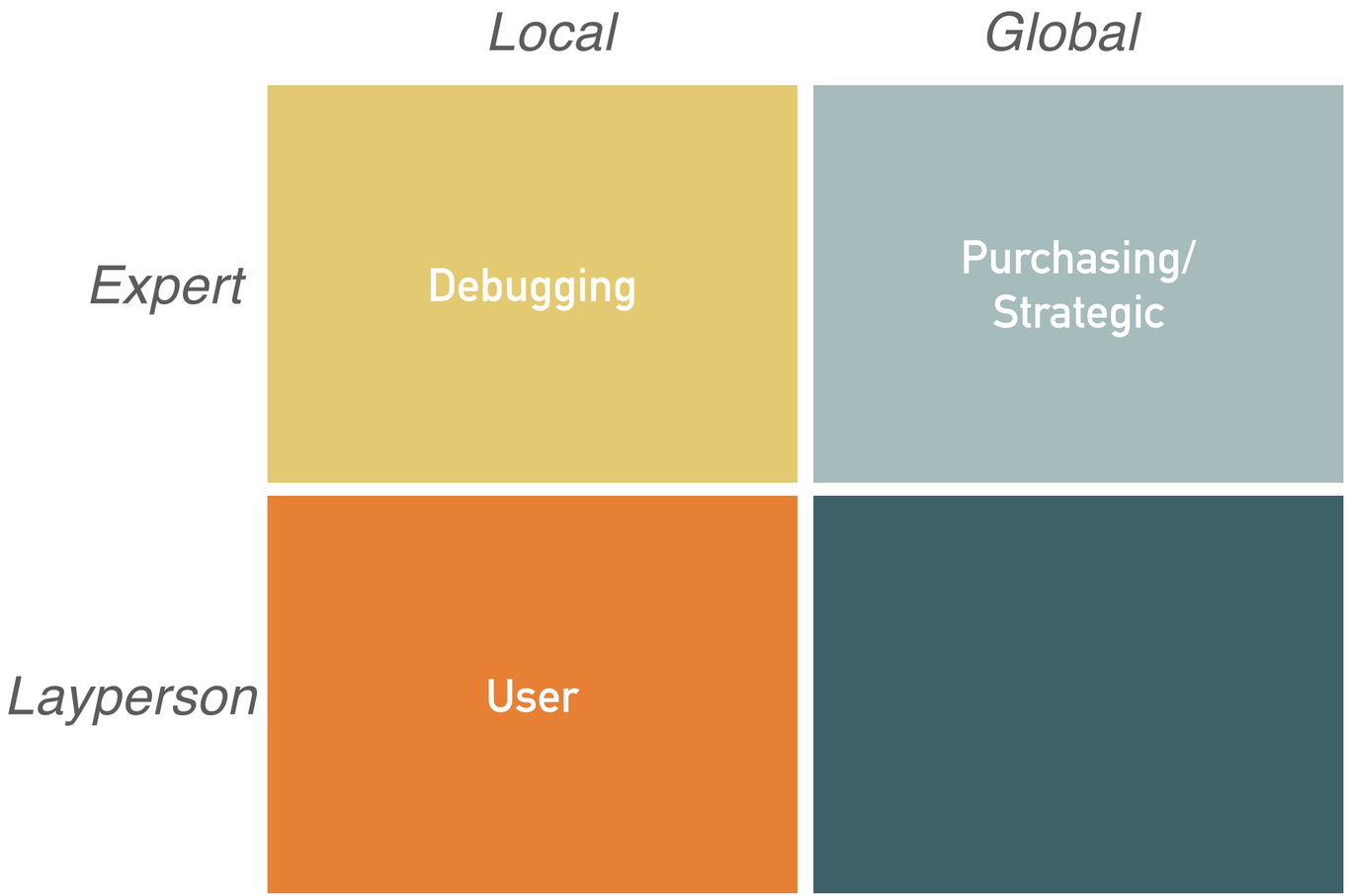
Tailored Explanations



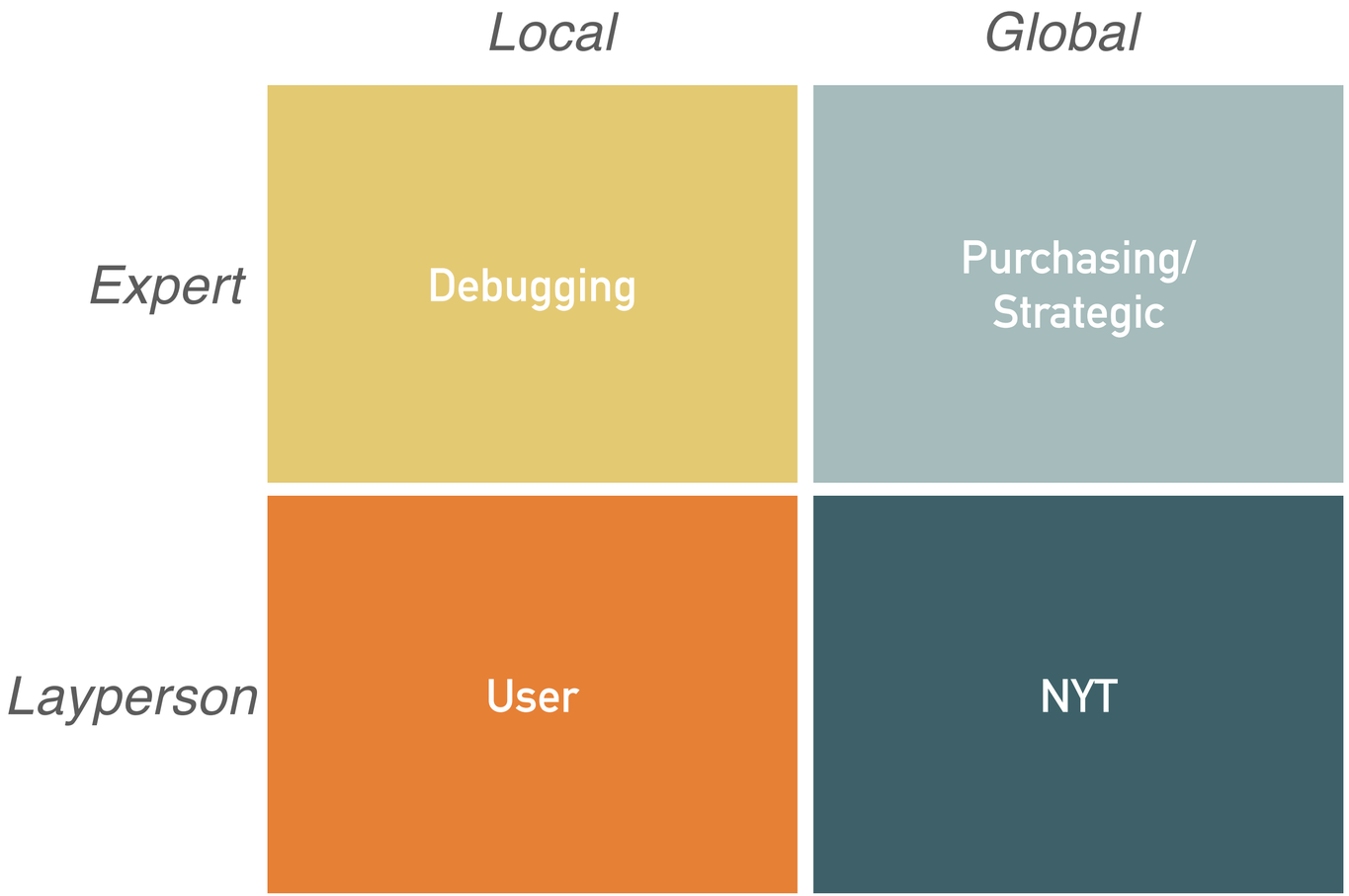
Tailored Explanations



Tailored Explanations



Tailored Explanations



What-if Explorations

Probe the model on various What-If scenarios.

- Examples:
  - What if “he” was replaced with “she”
  - What if we add a punctuation at the end of the sentence
- **Intuitive:** What you see is what you get
- **Highly expressive:** Most explainability techniques are a summarization of what-if behavior

Applications:

- Model understanding / debugging
- Algorithmic Recourse
- Prompt design

Debugging LMs

A Common Situation

- You've implemented an NLP system based on neural networks
- You've looked at the code, and it looks OK
- It has low accuracy, or makes incomprehensible errors
- **What do I do?**

Dimensions of Model Understanding

- **Debugging Implementation:** Identifying problems in your implementation (or assumptions)
- **Actionable Evaluation:** Identifying typical error cases and understanding how to fix them
- **Interpreting Predictions:** Examining individual predictions to dig deeper

LM Debugging is Paramount

- Models are often **complicated and opaque**
- **Everything is a hyperparameter** (network size, model variations, batch size/strategy, optimizer/ learning rate)
- Non-convex, stochastic optimization has **no guarantee of decreasing/converging loss**

```
@GPT: Due to the complexity of non-convex functions and the inherent
randomness of stochastic optimization methods, the optimization
results may be unstable, and in some cases, it may be impossible to
find a global optimum
```

Possible Causes

- **Training time problems**
  - Lack of model capacity
  - Poor training algorithm
  - Training time bug
- **Test time problems**
  - Disconnect between training and test
  - Failure of search algorithm
- **Overfitting**
- **Mismatch between optimized function and eval**

Don't debug all at once! Start top and work down.

Identifying Training Time Problems

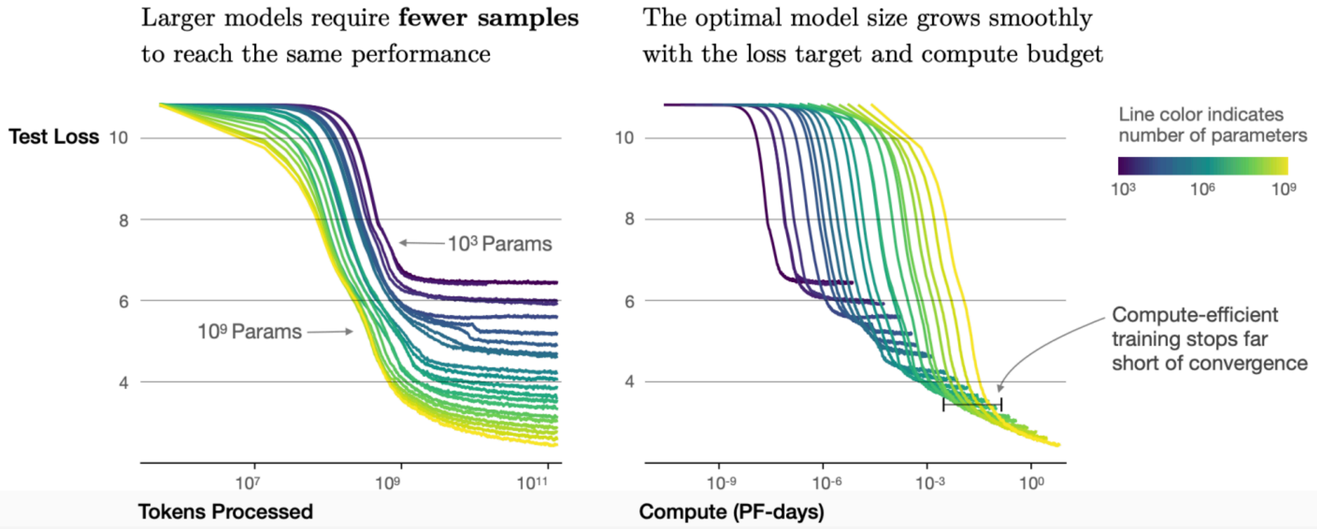
- Look at the **loss function** calculated on the **training set**
  - Is the loss function going down?
  - Is it going down basically to zero if you run training long enough (e.g. 20-30 epochs)?
  - If not, does it go down to zero if you use very small datasets?

Is my Model Too Weak?

- Larger models tend to perform better, esp. when pre-trained (e.g. Raffel et al. 2020)

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**90.3**	**71.6**	**97.5**	92.8	90.4	**93.1**	**92.8**

- Larger models can learn with fewer steps (Kaplan et al. 2020, Li et al. 2020)



Optimization Issues

- If increasing model size doesn’t help, you may have an optimization problem
- Check your
  - **optimizer** (an Adam variant is standard)
  - **learning rate** (is the rate you're using standard, are you using decay?)
  - **initialization** (if from scratch, are you using a reasonable initialization range)
  - **minibatching** (are you using sufficiently large batches?)
- Pay attention to these details when replicating previous work

Training/Test Disconnects

- Usually your loss calculation and prediction will be implemented in different functions
- Especially true for structured prediction models (e.g. encoder-decoders)
- Like all software engineering: **duplicated code is a source of bugs**!
- Also, usually loss calculation is minibatched, generation not.

Debugging Minibatching

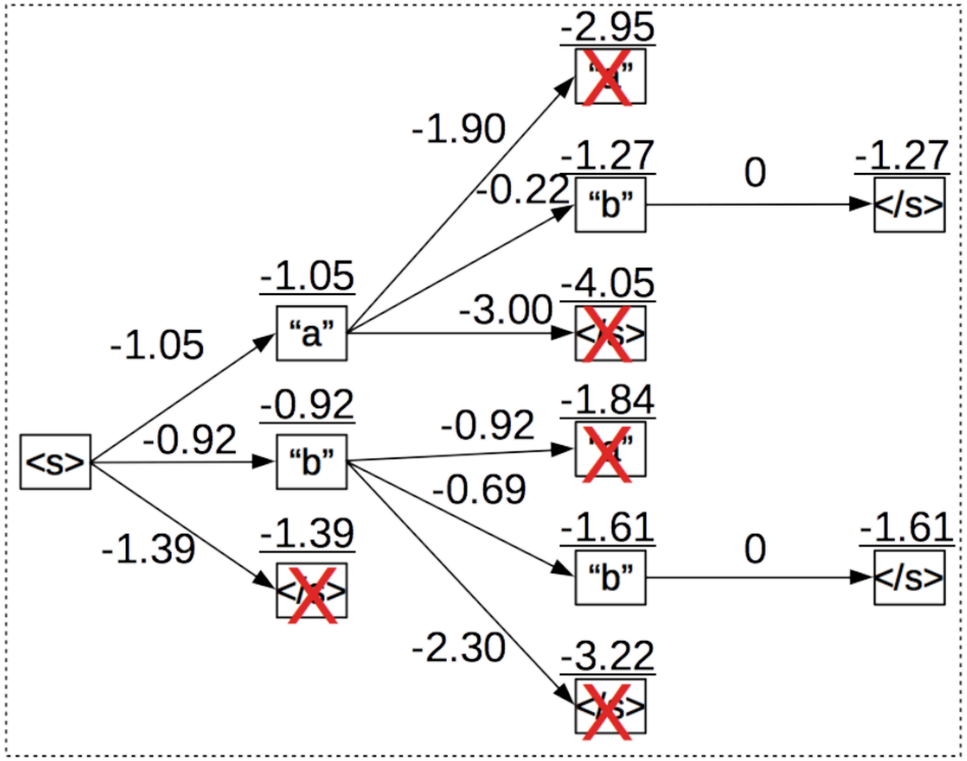
- Debugging mini-batched loss calculation
  - Calculate loss with **large batch size** (e.g. 32)
  - Calculate loss for **each sentence individually and sum**
  - The values should be the same (modulo numerical precision)
- Create a unit test that tests this!

Debugging Structured Generation

- Your decoding code should get the same score as loss calculation
- Test this:
  - Call **decoding function**, to generate an output, and keep track of its score
  - Call **loss function** on the generated output
  - The score of the two functions should be the same
- Create a unit test doing this!

Beam Search

- Instead of picking one high-probability word, maintain several paths



Debugging Search

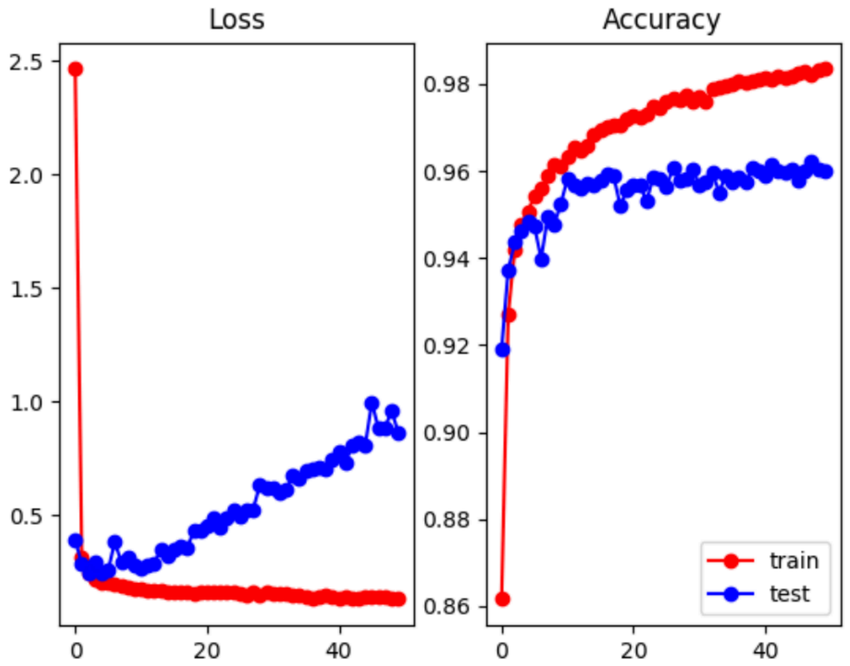
- As you make search better, the **model score** should get better (almost all the time)
- Search w/ varying beam sizes and make sure you get a better overall model score with larger sizes
- Create a unit test testing this!

Mismatch Between Loss and Evaluation Metric

- It is very common to optimize for maximum likelihood for training
- But even though likelihood is getting better, accuracy can get worse

Example: Classification

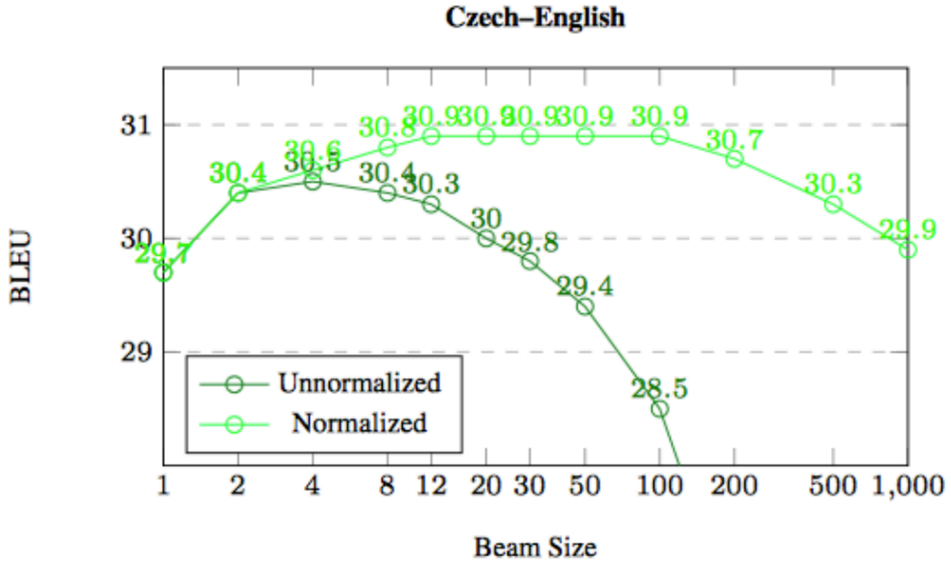
- Loss and accuracy are de-correlated (see dev)

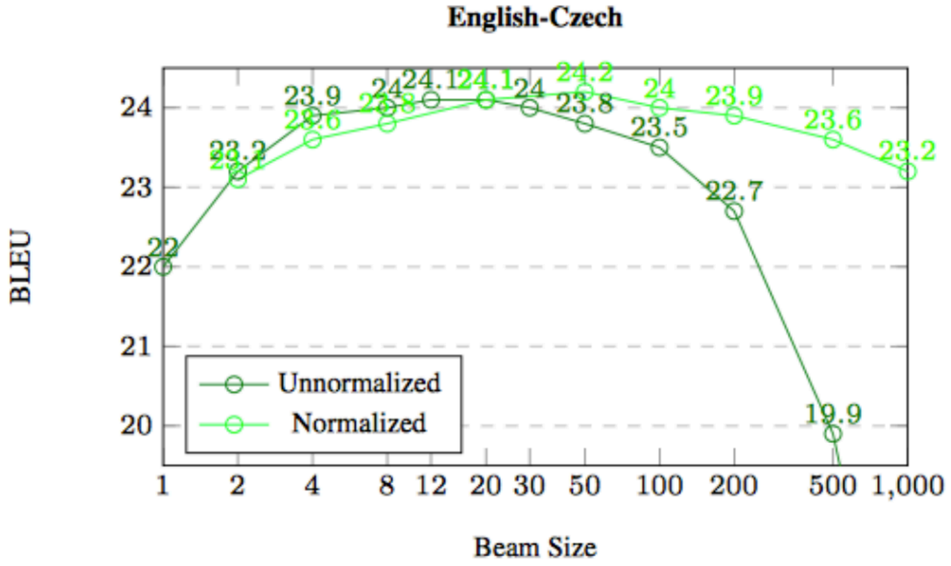


- Why? Model gets more confident about its mistakes.

A Starker Example (Koehn & Knowles 2017)

- Better search (=better model score) can result in worse BLEU score!

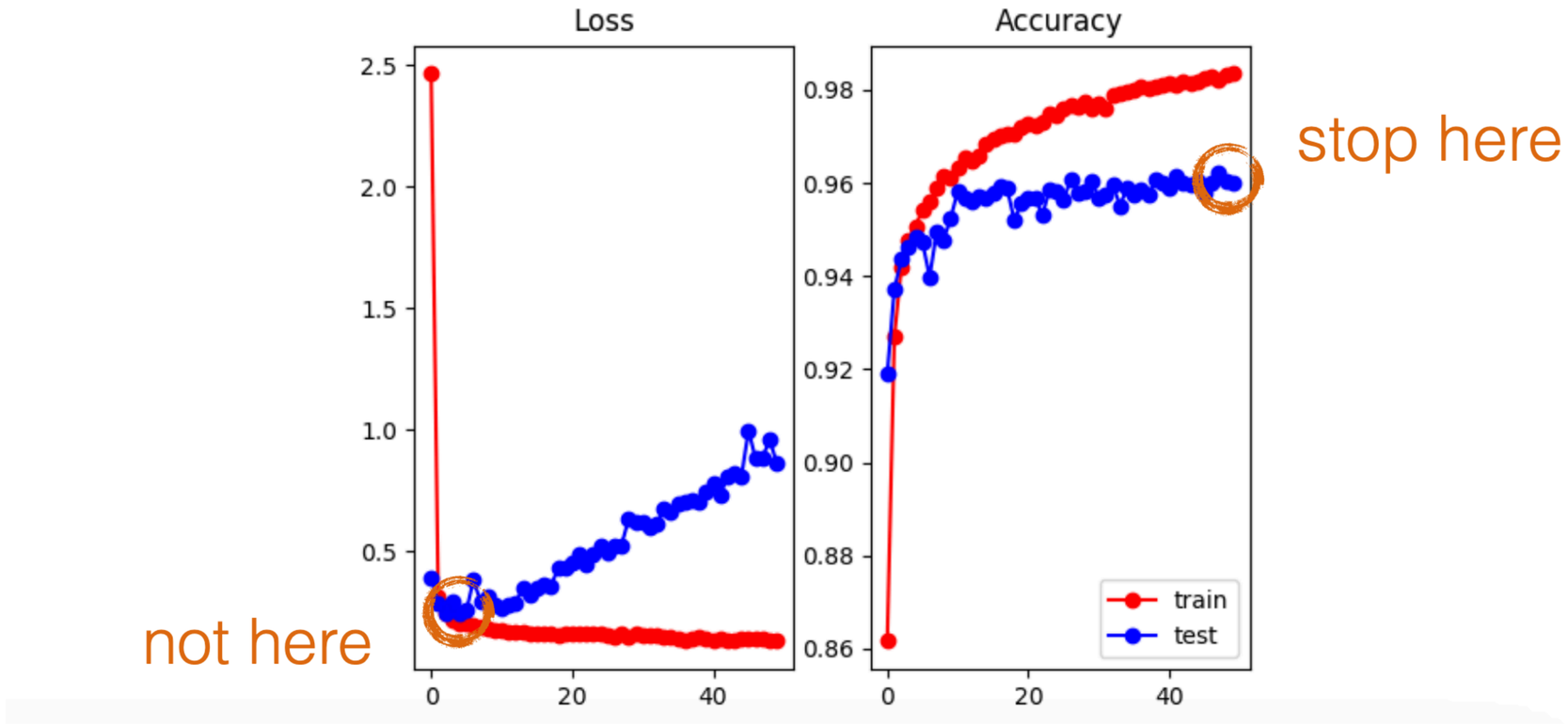




- Why? Shorter sentences have higher likelihood, better search finds them, but BLEU likes correct-length sentences.

Managing Loss/Metric Differences

- Most principled way: use a method like reinforcement learning
- Easier way: Early stopping w/ evaluation metric

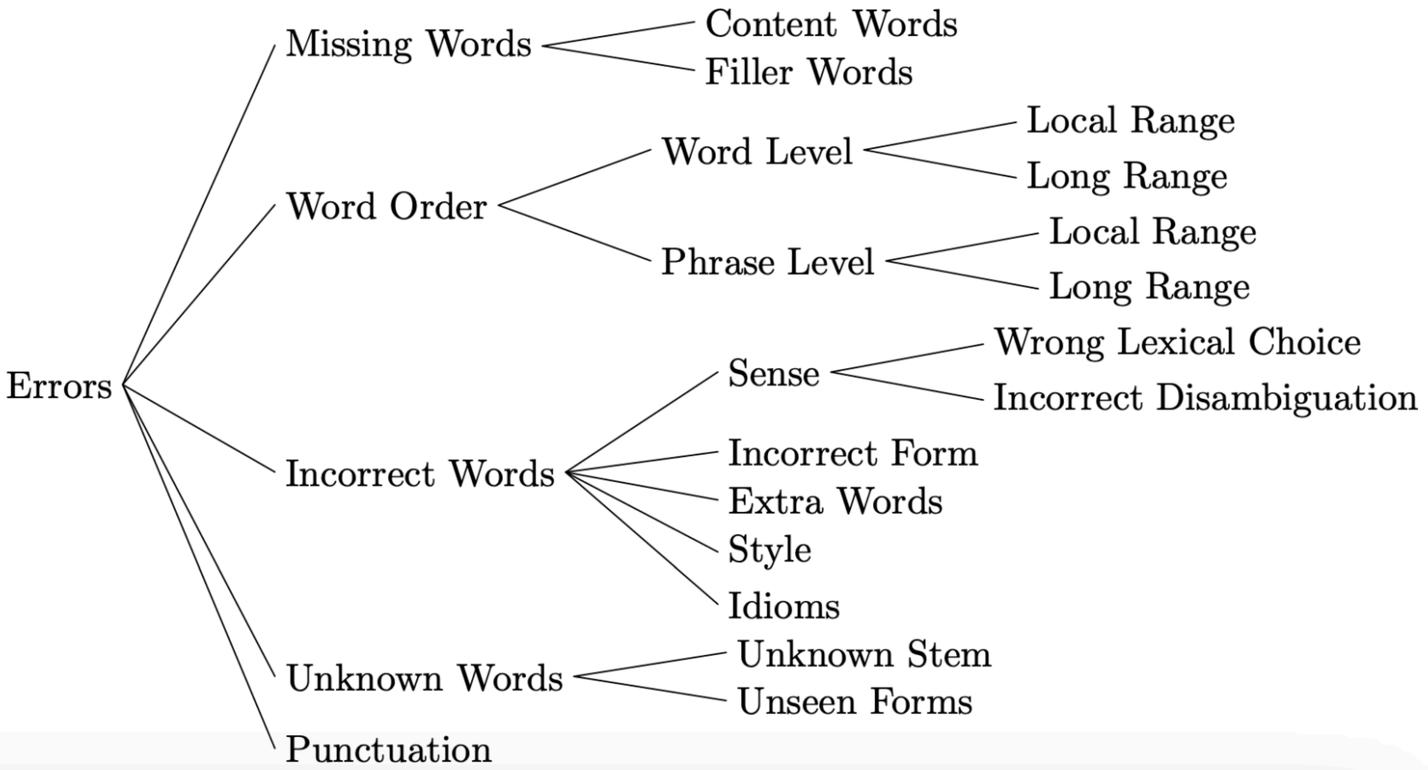


Look at Your Data!

- Both bugs and research directions can be found by **looking at your model outputs**
- The first word of the sentence is dropped every generation
  > went to the store yesterday
  > bought a dog
  → implementation error?
- The model is consistently failing on named entities
  → need a better model of named entities?

Systematic Qualitative Error Analysis

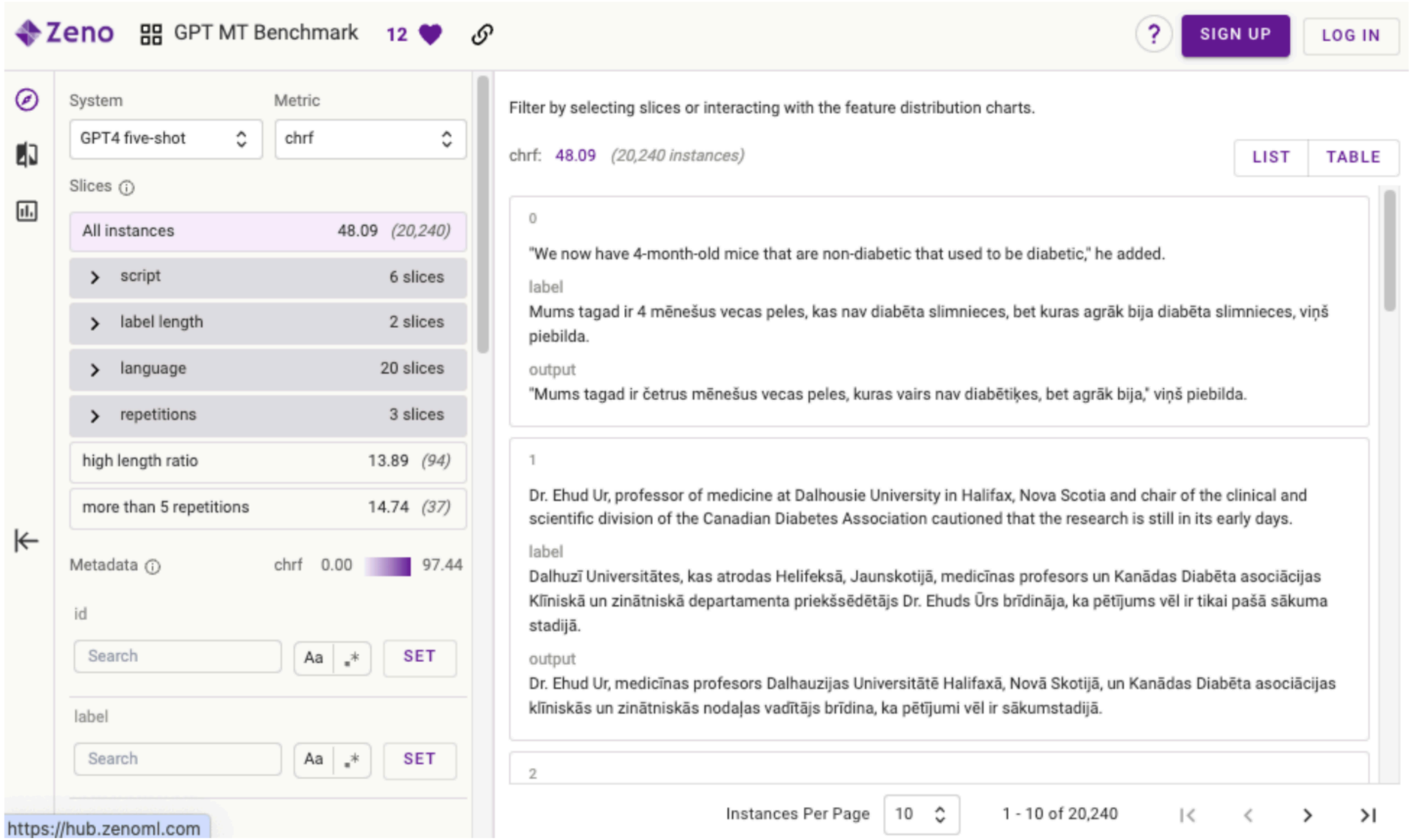
- **Look at 100-200 errors**
- Try to **group them** into a typology (pre-defined or on the fly)
- Example: Vilar et al. (2006)



Quantitative Error Analysis

- Measure gains quantitatively. What is the phenomenon you chose to focus on? Is that phenomenon getting better?
  - **You focused on low-frequency words:** is accuracy on low frequency words increasing?
  - **You focused on syntax:** is syntax or word ordering getting better, are you doing better on long-distance dependencies?
  - **You focused on search:** how many search errors are being reduced?

Example: Zeno



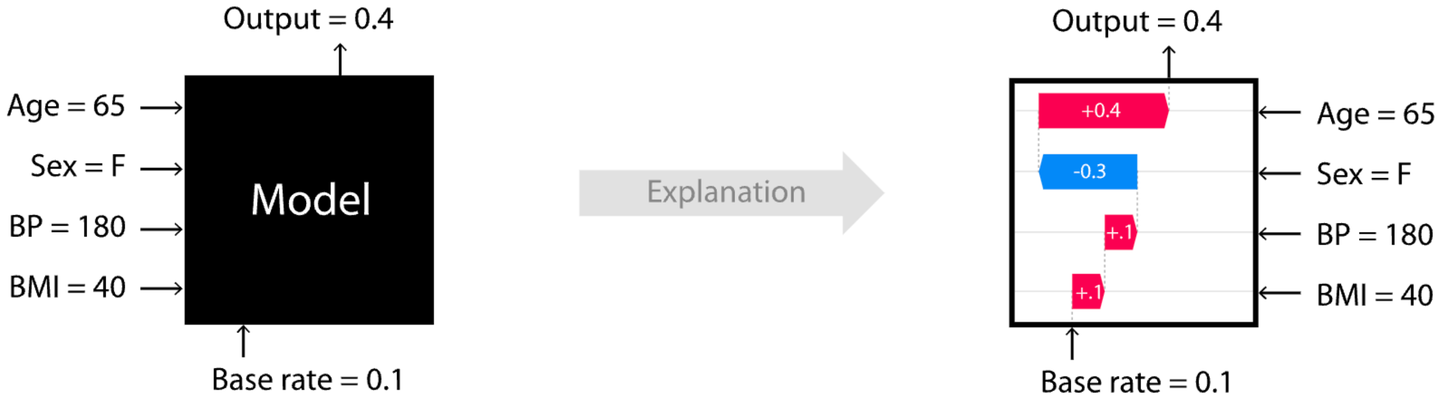
https://zenoml.com

Shapley Values

Shapley Values

- Game-theoretic analyses of feature importance
- **How much** and **in which direction** do permutations affect outcomes **per instance**?

SHAP



Origin

- Classic result in game theory on distributing the total gain from a **cooperative game**
- Introduced by **Lloyd Shapley** in **1953**[1], who later won the **Nobel Prize in Economics** in the 2012
- Popular tool in studying cost-sharing, market analytics, voting power, and most recently **explaining ML models**

Lloyd Shapley in 1980

[1] **"A Value for n-person Games"**. Contributions to the Theory of Games 2.28 (1953): 307-317

Intuition

- Players {1, …, M} collaborating to generate some **gain**
  - Think: Employees in a company creating some profit
  - Described by a **set function v(S)** specifying the gain for any subset $S \subseteq \{1, \ldots, M\}$

- **Shapley values** are a fair way to attribute the total gain to the players
  - Think: Bonus allocation to the employees
  - Shapley values are commensurate with the player's contribution

Formal Notion

$$\phi_i(v) = \mathop{\mathbb{E}}_{\boldsymbol{O} \sim \pi(M)} [v(\text{pre}_i(\boldsymbol{O}) \cup \{i\}) - v(\text{pre}_i(\boldsymbol{O}))]$$

- Consider all possible permutations $\pi(M)$ of players (**M! possibilities**)
- In each permutation $\boldsymbol{O} \sim \pi(M)$
  - Add players to the coalition in that order
  - Note the marginal contribution of each player i to set of players before it in the permutation, i.e., $v(pre_i(\boldsymbol{O}) \cup \{i\}) - v(pre_i(\boldsymbol{O}))$
- The average marginal contribution across all permutations is the Shapley Value

An Example

A company with two employees **Alice** and **Bob**

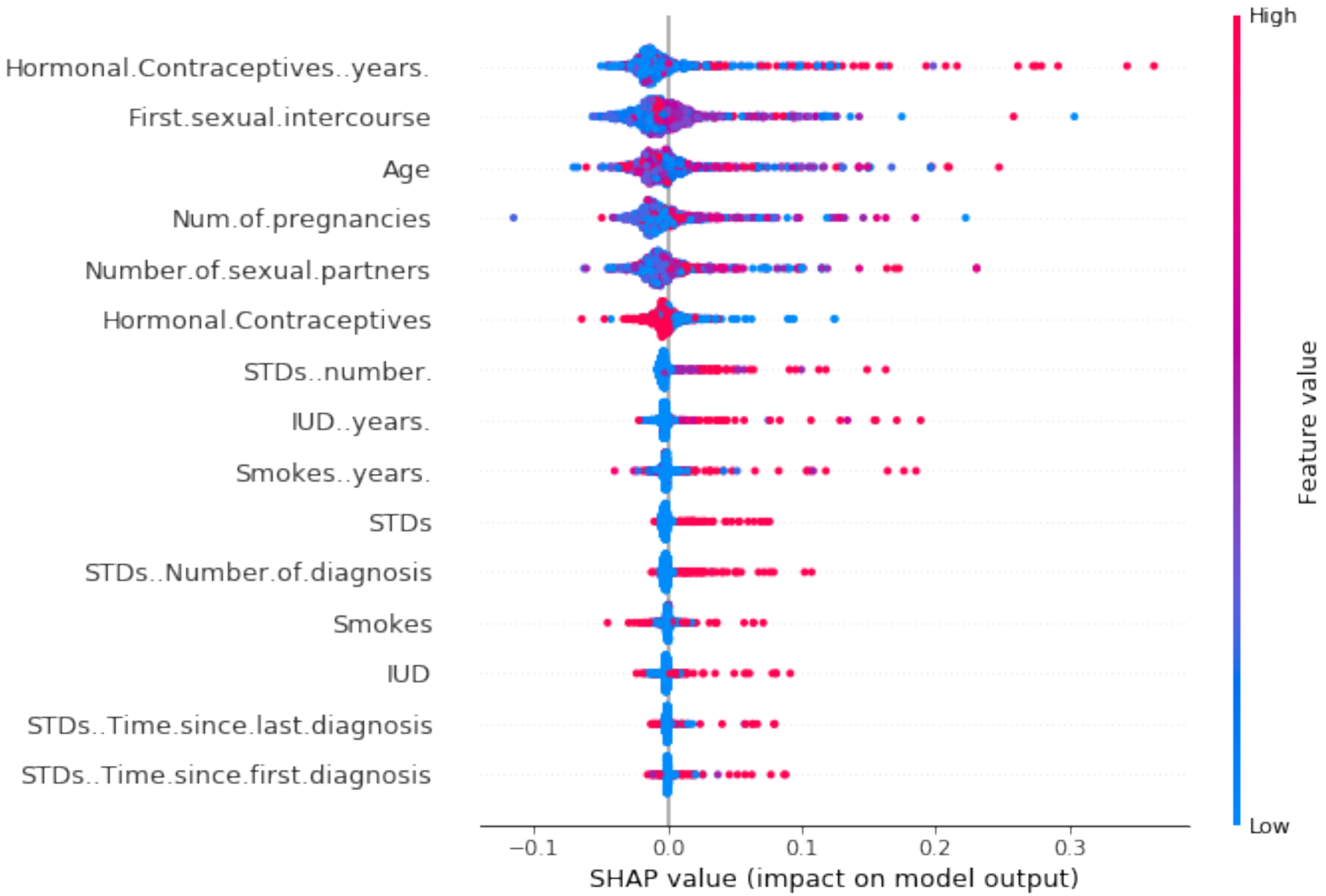
- No employees, no profit [v({}) = 0]
- Alice alone makes 20 units of profit [v({Alice}) = 20]
- Bob alone makes 10 units of profit [v({Bob}) = 10]
- Alice and Bob make 50 units of profit [v({Alice, Bob}) = 50]

What should the bonuses be?

Permutation	**Marginal for Alice**	**Marginal for Bob**
Alice, Bob	20	30
Bob, Alice	40	10
Shapley Value	**30**	**20**

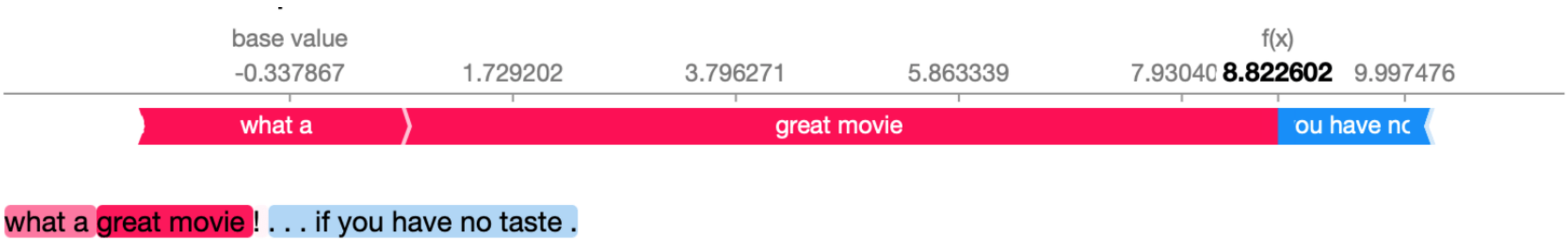
Going Global

- Generally a local method
- But can be cumulated across examples



SHAP for NLP

- Tokens as players
- **positive** vs. **negative** affect



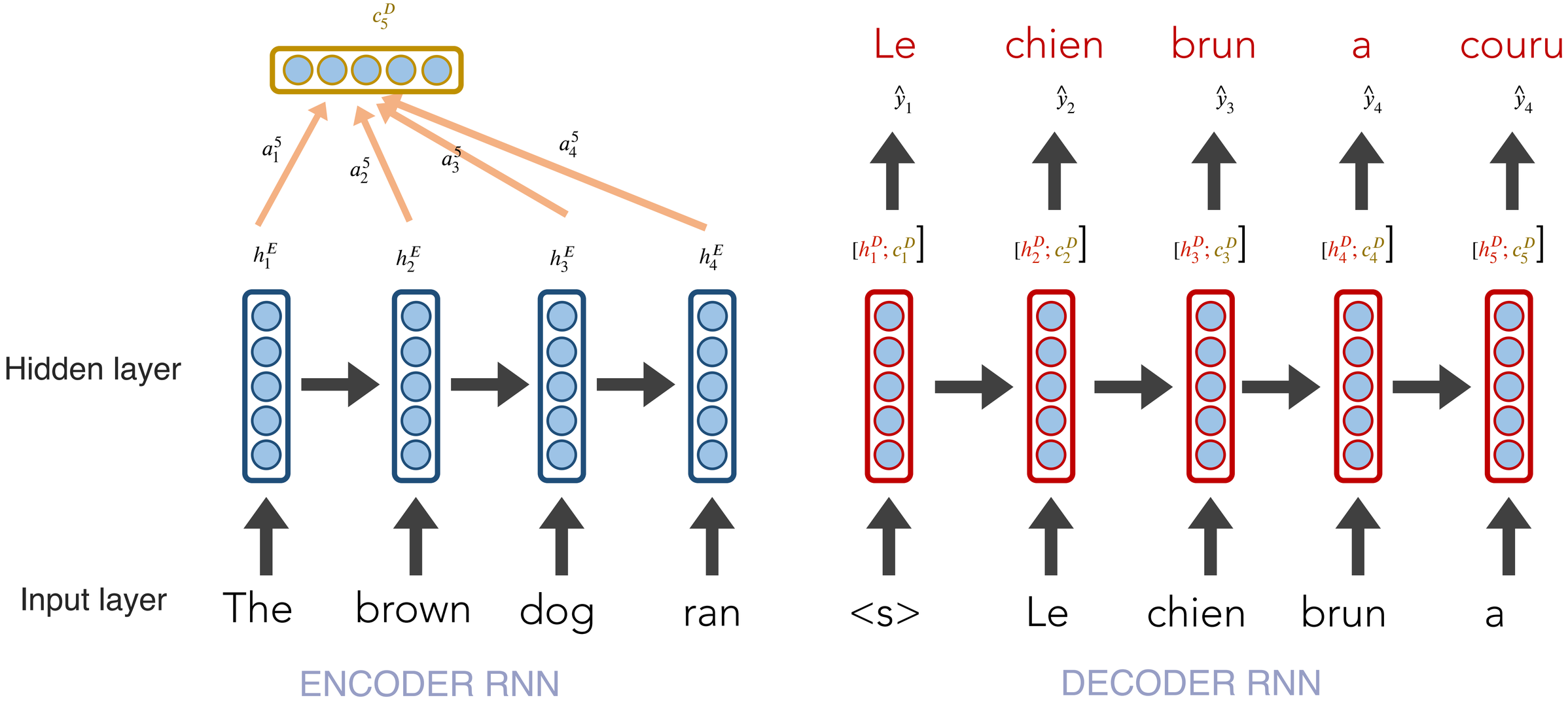
SHAP Summary

- **Advantages**
  - Sound theoretical foundation
  - Model agnostic (to the point that you might not need params!)
  - Directional
- **Disadvantages**
  - **Factorial Complexity!** (Unless you fix it)
  - No mechanistic understanding

Attention Visualization

Attention Recap

REMEMBER: each attention weight a_i^j is based on the decoder's current hidden state, too.

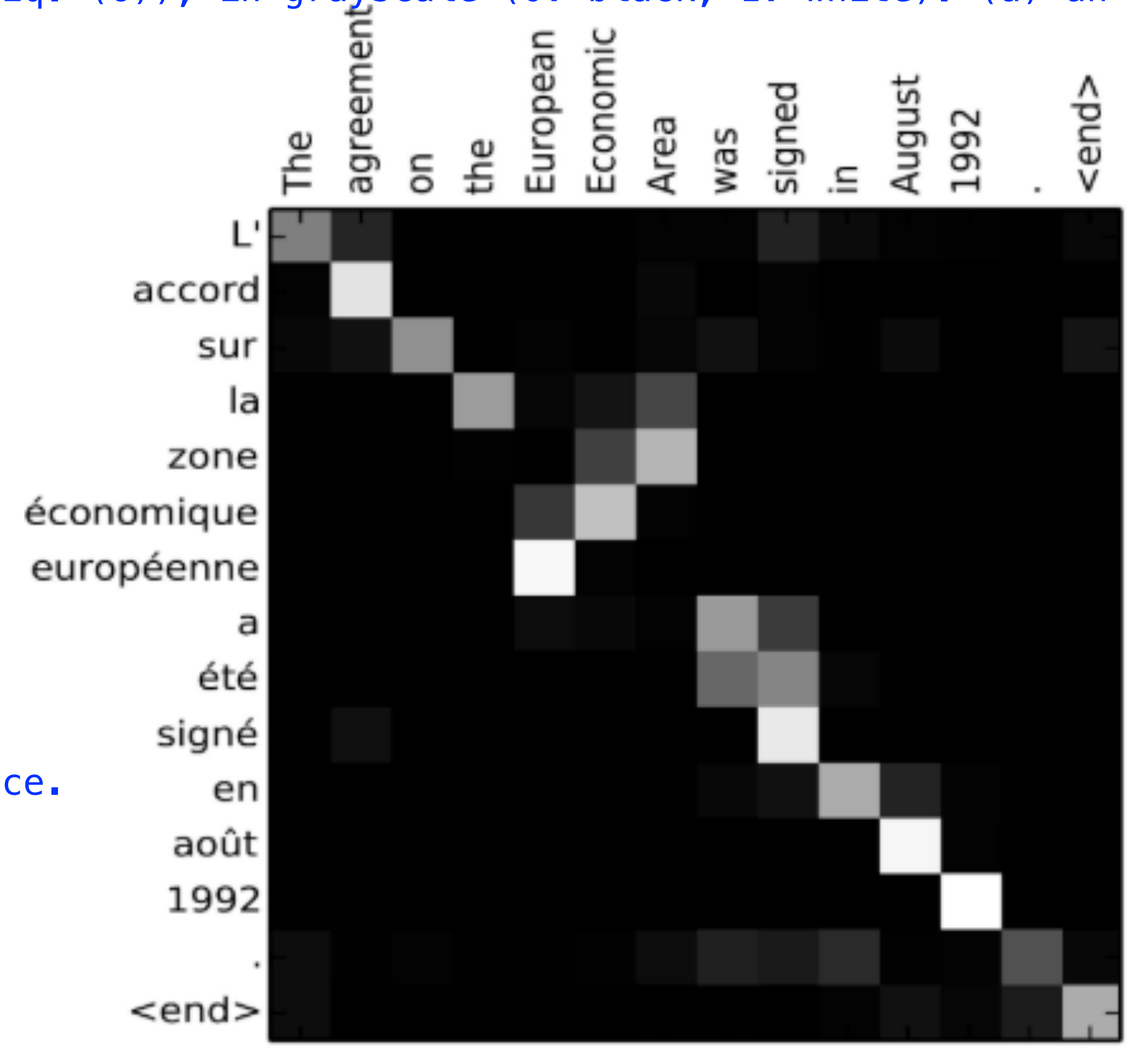


Attention Recap

The x-axis and y-axis of each plot
correspond to the words in the source sentence (English) and
the generated translation (French), respectively.
Each pixel shows the weight αij of the annotation of the j-th source word for
the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an
arbitrary sentence.

Attention:

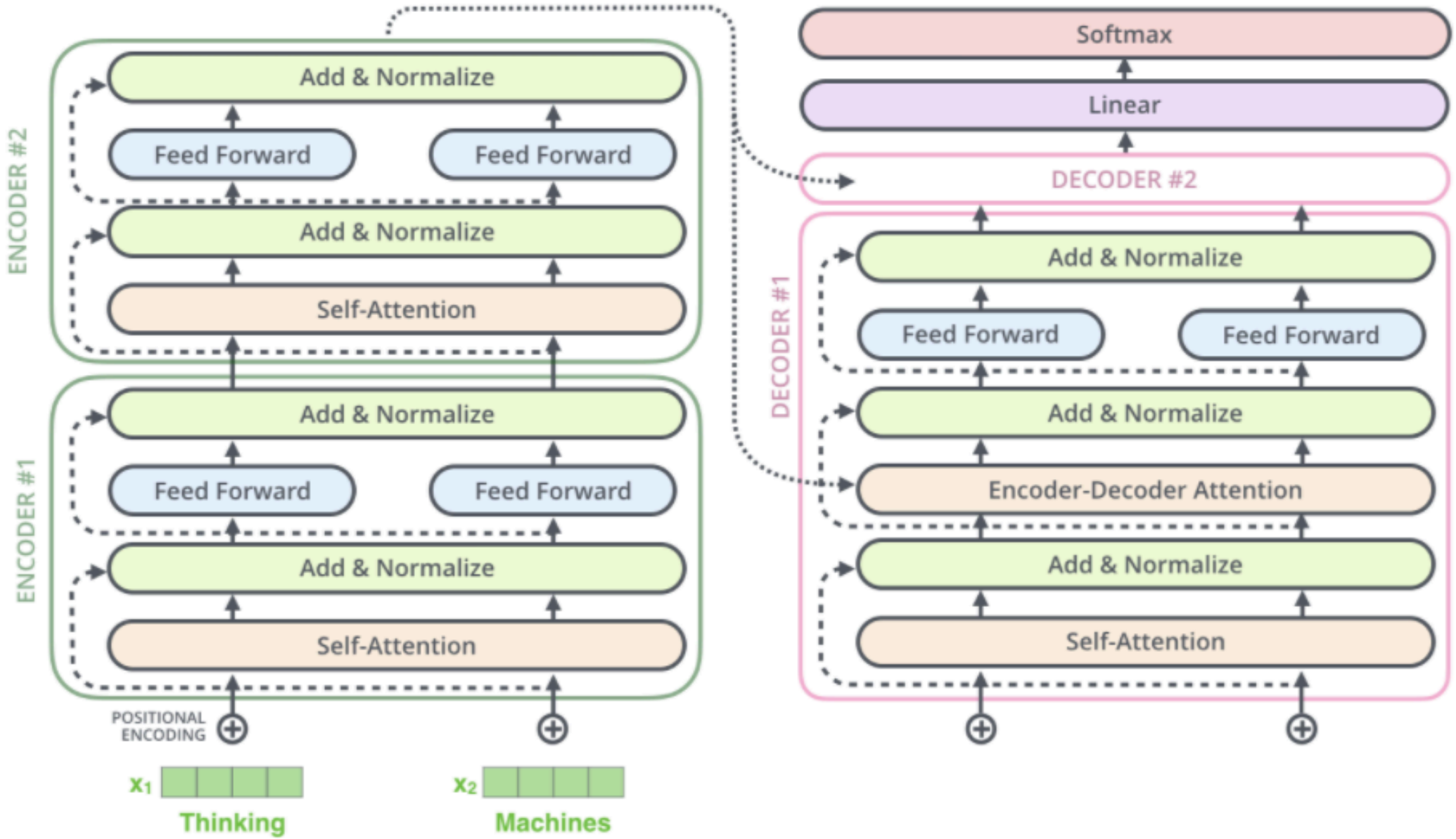
- greatly improves seq2seq results
- allows us to visualize the contribution each encoding word gave for each decoder's word



to inspect the (soft-)alignment between the words
in a generated translation and those in a source sentence.
This is done by visualising the annotation
weights αij from Eq. (6), as in Fig. 3.
Each row of a matrix in each plot indicates the weights
associated with the annotations.

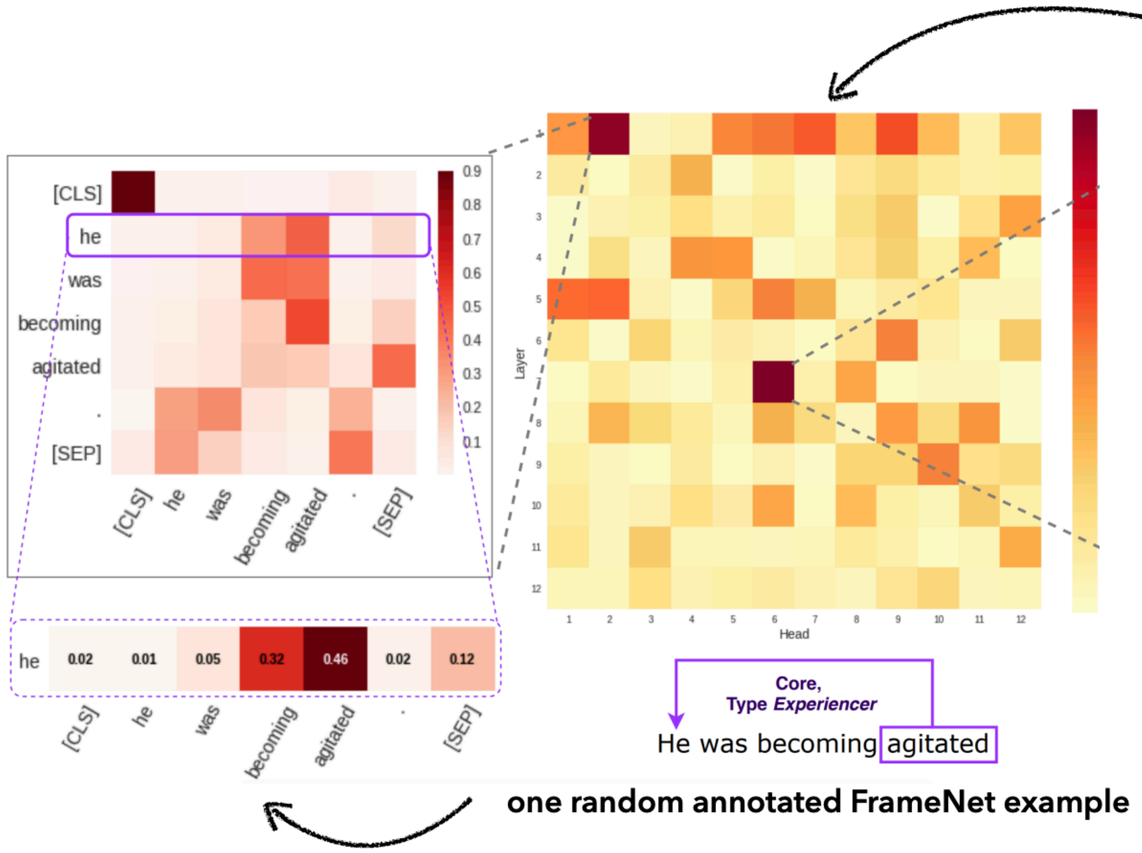
Image source: Fig 3 in [Bahdanau et al., 2015]

Where to Look?



“Interpretable” Attention Heads

– Relation-specific heads



HEAD IMPORTANCE

average max absolute attention weight among FrameNet-annotated tokens

Voita et al, 2019:
Layerwise Relevance Propagation (LRP)

Michel et al, 2019:

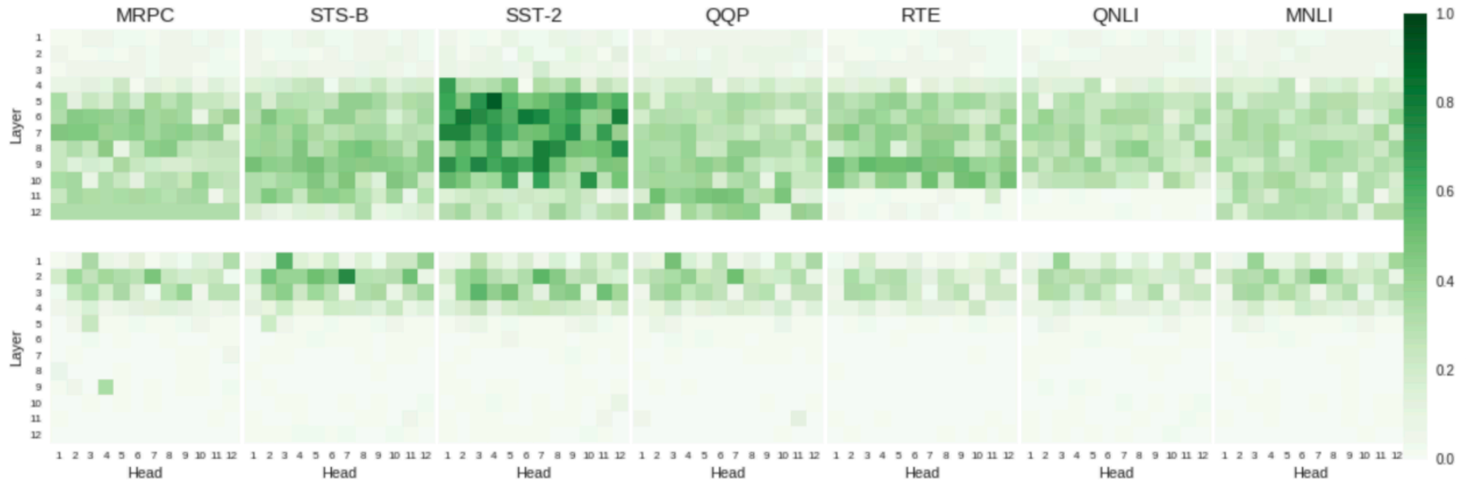
$$I_h = \mathbb{E}_{x \sim X} \left| A_{h,l}(x)^T \frac{\partial \mathcal{L}(x)}{\partial A_{h,l}(x)} \right|$$

one random annotated FrameNet example

“Interpretable Attention Heads”

Hypothesis: task-finetuning of BERT creates patterns reflecting linguistic features, i.e. specific tokens get higher attention weights, producing **vertical stripes** on the corresponding attention maps

✗ **task-finetuning creates patterns associated with the [SEP] and [CLS] tokens**



Attention is [not]* Explanation

Wiegreffe and Pinter, 2019

MAYBE

- adversarial attentions weights can be learned
- ∃ tests to determine when/whether attention can be used as an explanation

Jain and Wallace, 2019

NO

- attention weights don't correlate with leave-one-out methods
- ∃ alternative attention weights with near-identical predictions

Moradi et al, 2019

NO

- ∃ alternative attention weights with near-identical predictions

Should attention be treated as justification for a prediction?

Serrano and Smith, 2019

NO

- analysis based on intermediate representation erasure shows that attention weights often fail to identify representations most important to the model's final decision

Zhong et al, 2019

MAYBE

- attention was often misaligned with the words that contribute to sentiment
- attention trained with human rationales brings faithful explanations

Gradient Tracing

Visual Question Answering

Q. How symmetric are the white bricks on either side of the building?

Model answers: very
Ground truth: very

Thoughtfully constructed training data

200K images, 600K questions

Test accuracy of Kazemi and Elqursh (2017) model: **61%**

Naive Approaches

- **Ablations**: Drop each feature and note the change in prediction
  - Computationally expensive, Unrealistic inputs, Misleading when features interact
- **Feature*Gradient**: Attribution for feature x_i is $x_i * \partial y/\partial x_i$

Prediction: **“fireboat”**

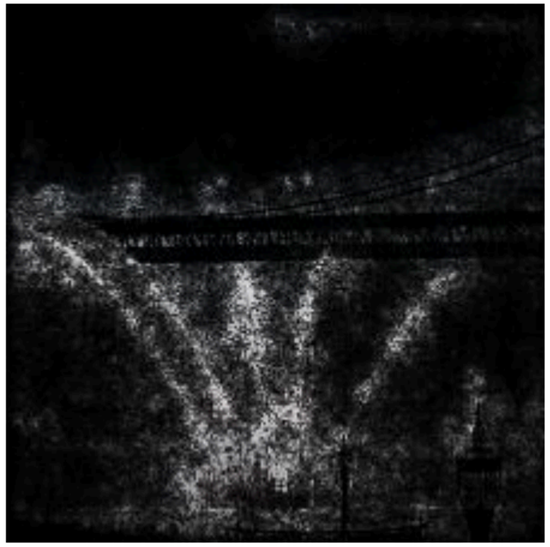
Integrated Gradients [ICML 2017]

Integrate the gradients along a **straight-line path from baseline to input**

$$IG(input, base) ::= (input - base) * \int_{0\ -1} \nabla F(\alpha * input + (1-\alpha) * base)\ d\alpha$$

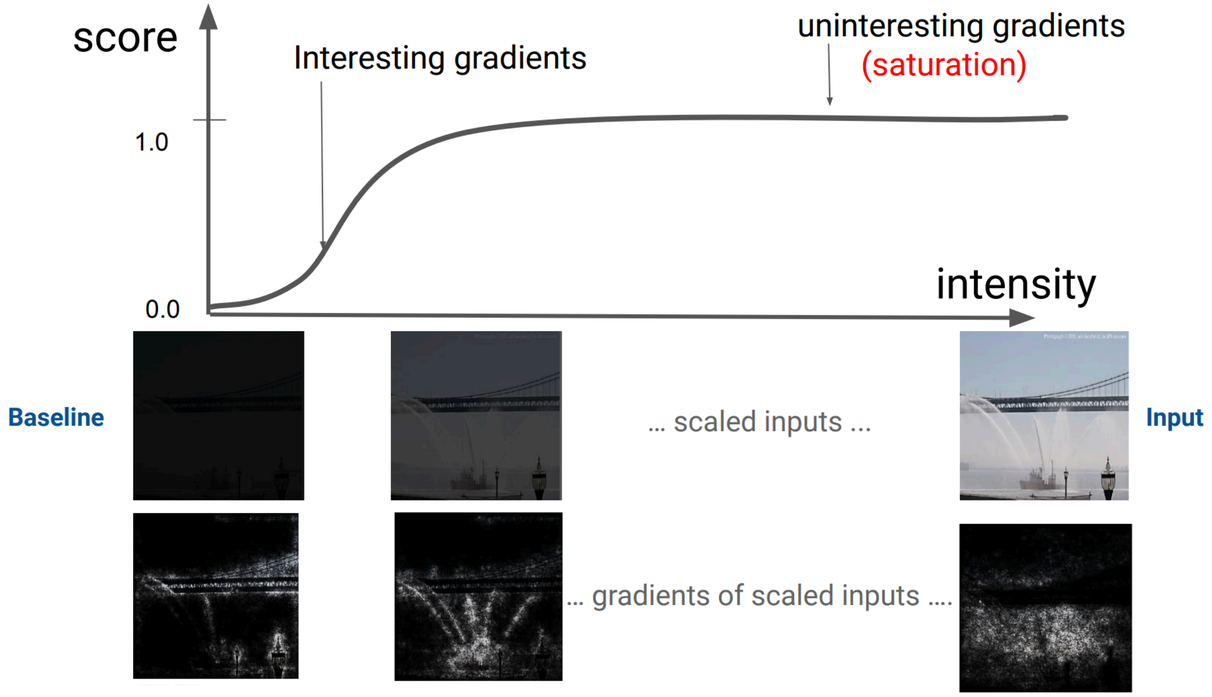
Original image

Integrated Gradients



Picking Your Baseline

- Ideally, the baseline is an **informationless input for the model**
  - E.g., Black image for image models
  - E.g., Empty text or zero embedding vector for text models
- **Integrated Gradients explains F(input) - F(baseline) in terms of input features**



Why is this Image labeled as a “clog”?

Original image

“Clog”

Why is this Image labeled as a “clog”?

Original image

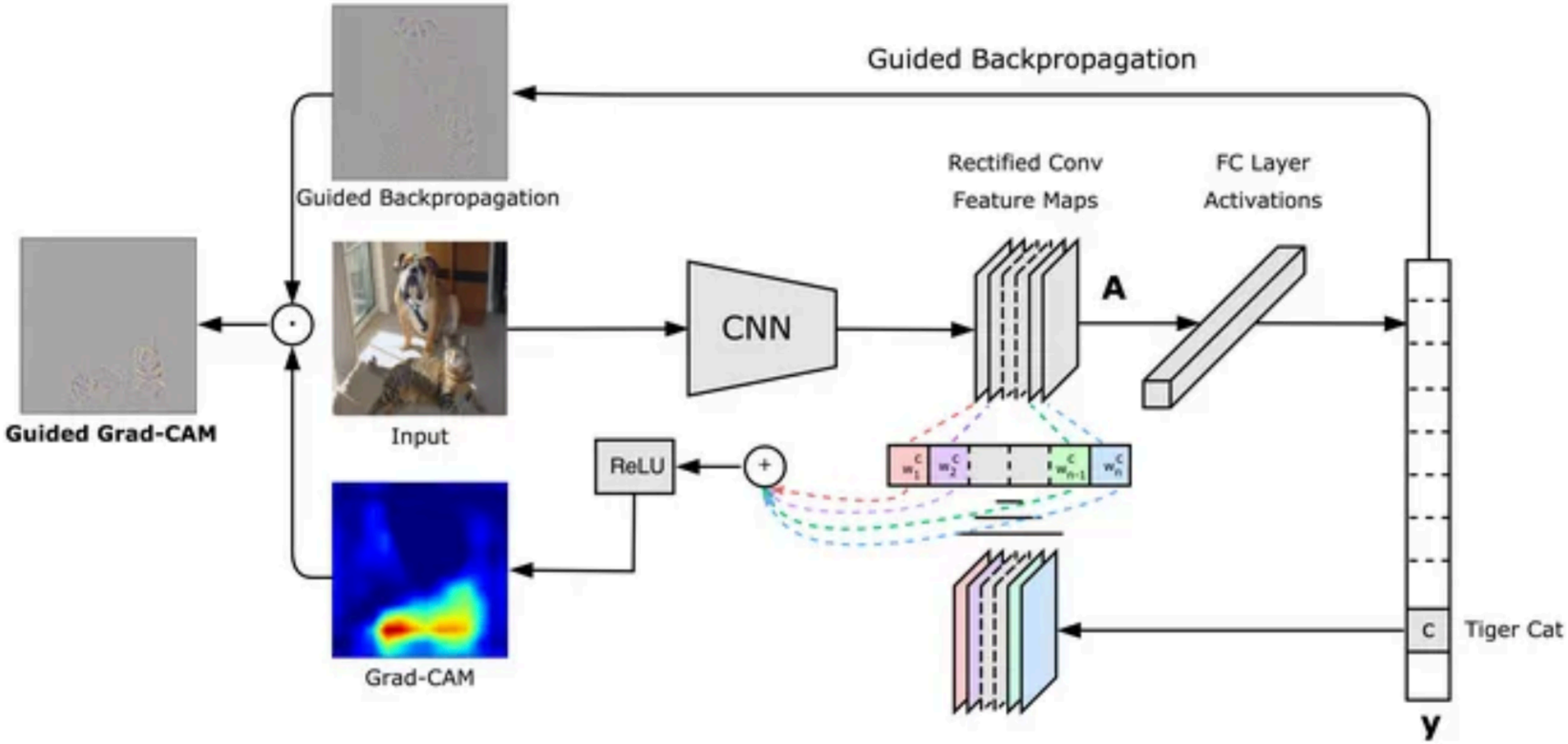
Integrated Gradients
(for label “clog”)

“Clog”

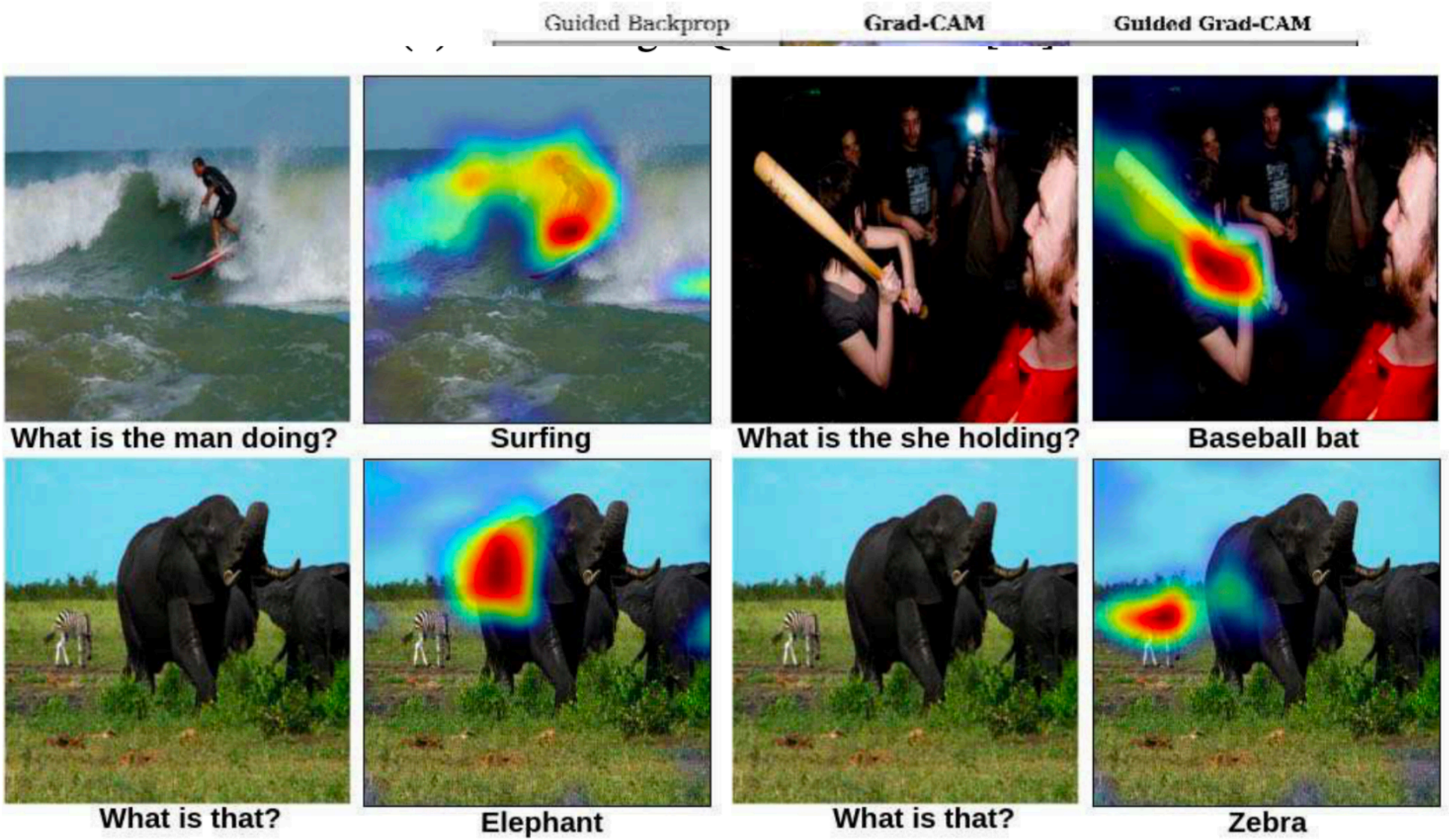
Next step: Gather more images of Clogs of different colors?

Class Activation Mapping (CAM)

Grad-CAM


Guided Backpropagation
Guided Backpropagation
Rectified Conv
Feature Maps
FC Layer
Activations
Guided Grad-CAM
Input
CNN
A
ReLU
Grad-CAM
C
Tiger Cat
y

Grad-CAM VQA


Guided Backprop
Grad-CAM
Guided Grad-CAM
What is the man doing?
Surfing
What is the she holding?
Baseball bat
What is that?
Elephant
What is that?
Zebra

Gradient Tracing Methods

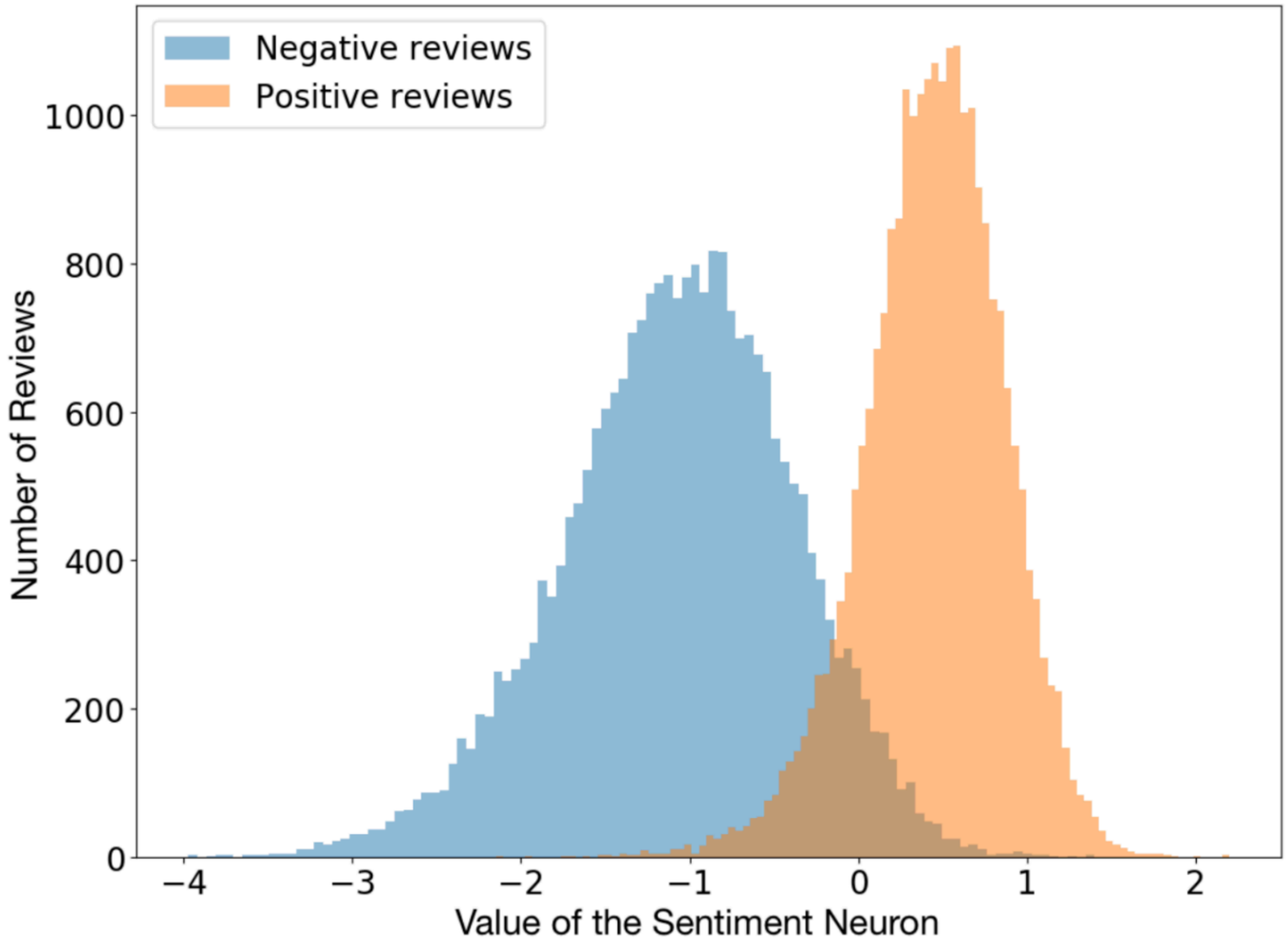
Gradient Tracing Summary

- **Advantages**
  - Model specific
  - Applicable to complex models
- **Disadvantages**
  - Input dependence
  - Gradient saturation

Probing

The Sentiment Neuron

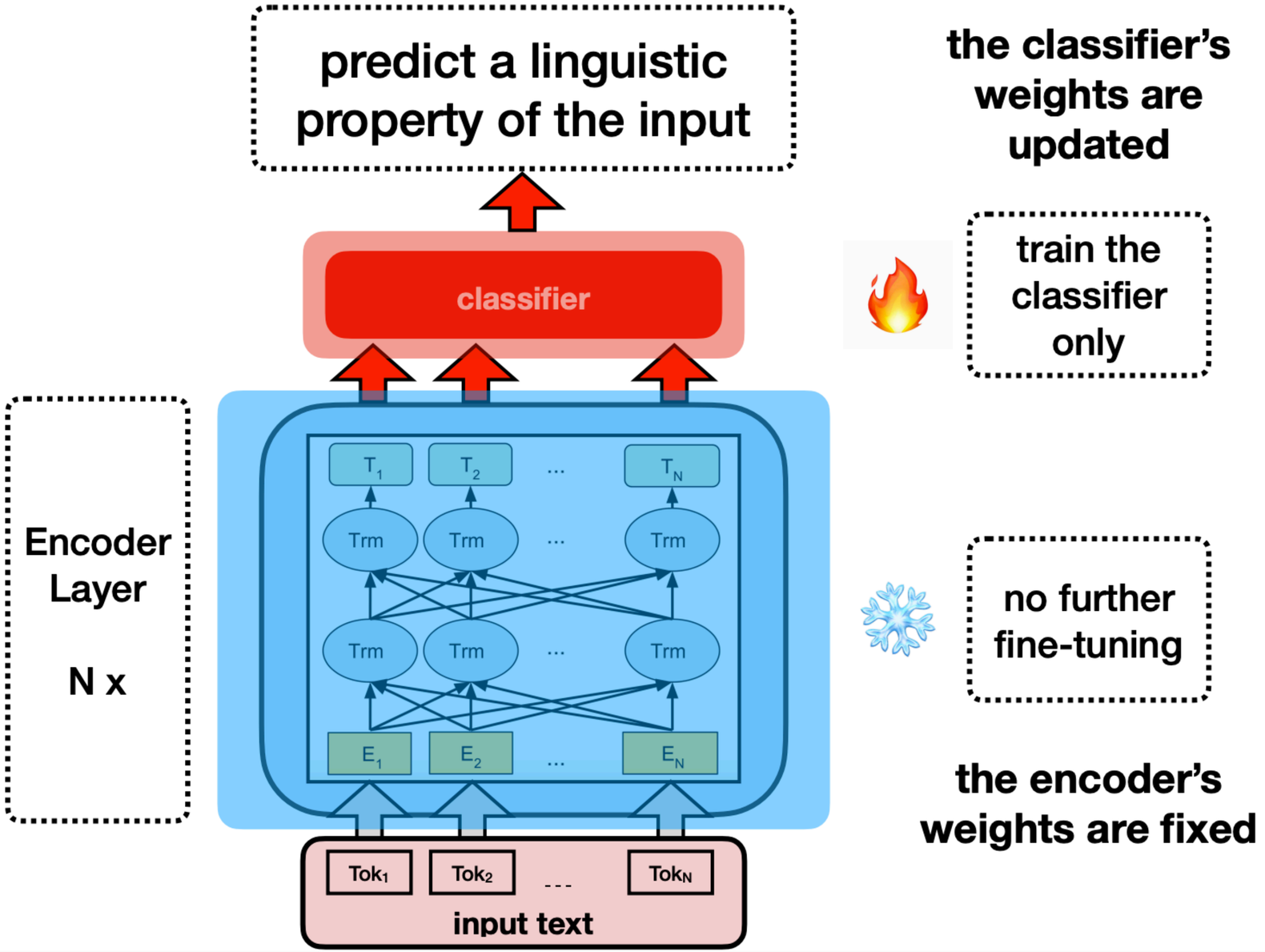
While training the linear model with L1 regularization, we noticed it used surprisingly few of the learned units. Digging in, we realized there actually existed a single “sentiment neuron” that’s highly predictive of the sentiment value.



Linguistic Probing Tasks

given an encoder model (e.g., BERT) pre-trained on a certain task, we use the representations it produces to train a classifier (without further fine-tuning the model) to predict a linguistic property of the input text

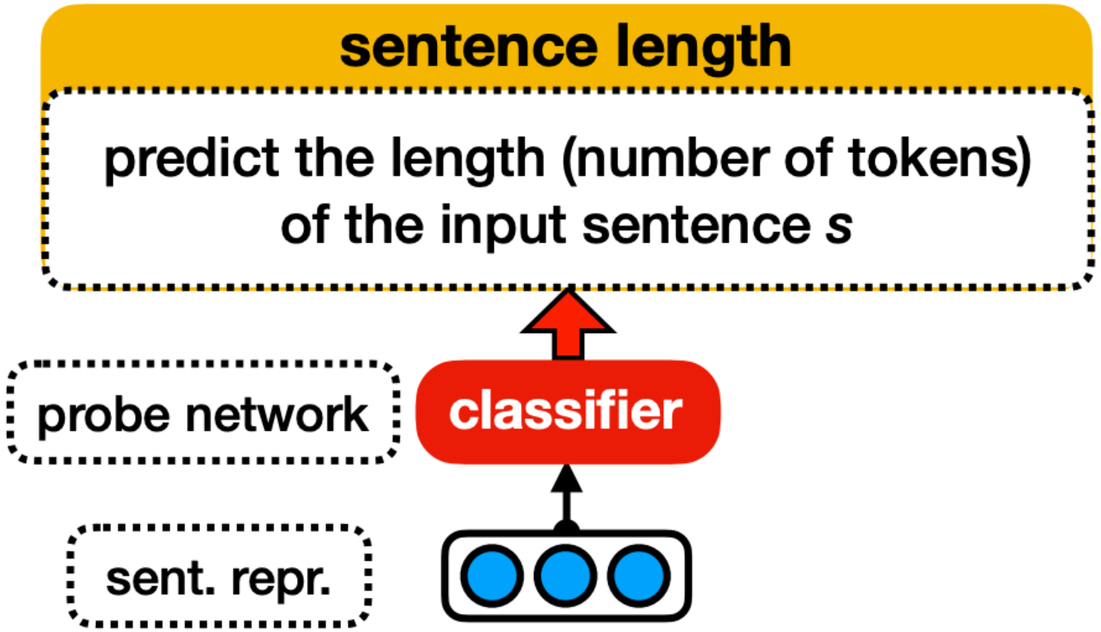
Probing Architecture



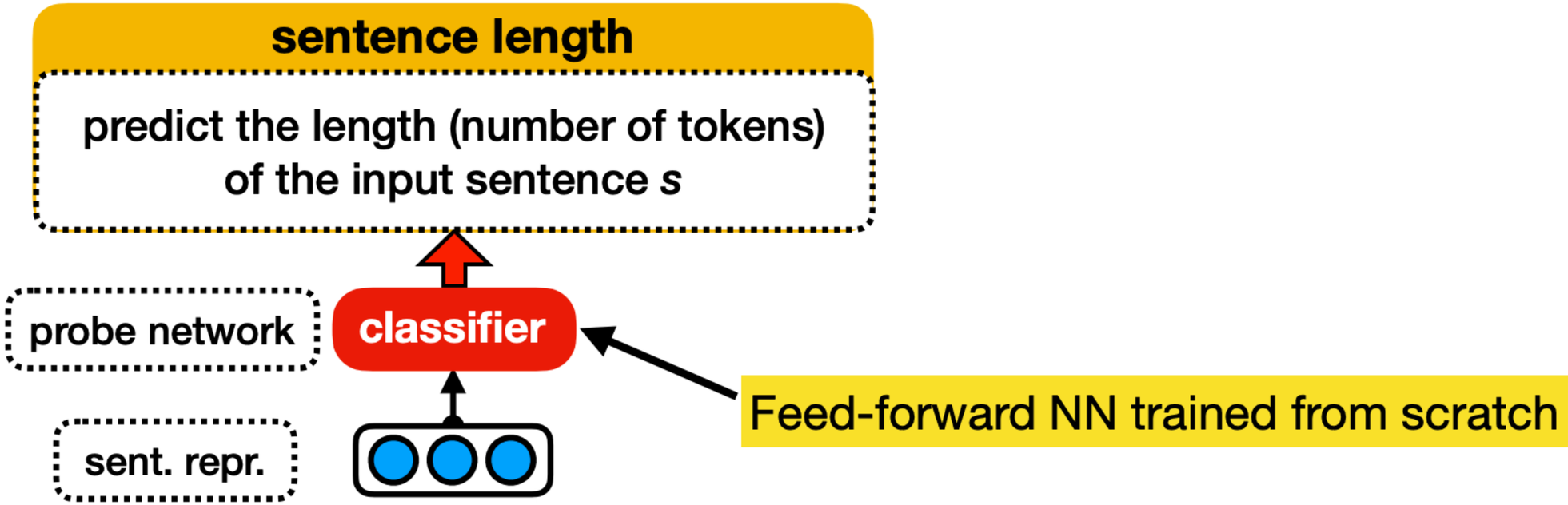
Probing Tasks

- usually classification problems that focus on simple linguistic properties
- ask simple questions, minimizing interpretability problems
- because of their simplicity, it is easier to control for biases in probing tasks than in downstream tasks
- the probing task methodology is agnostic with respect to the encoder architecture, as long as it produces a vector representation of input text
- does not necessarily correlate with downstream performance

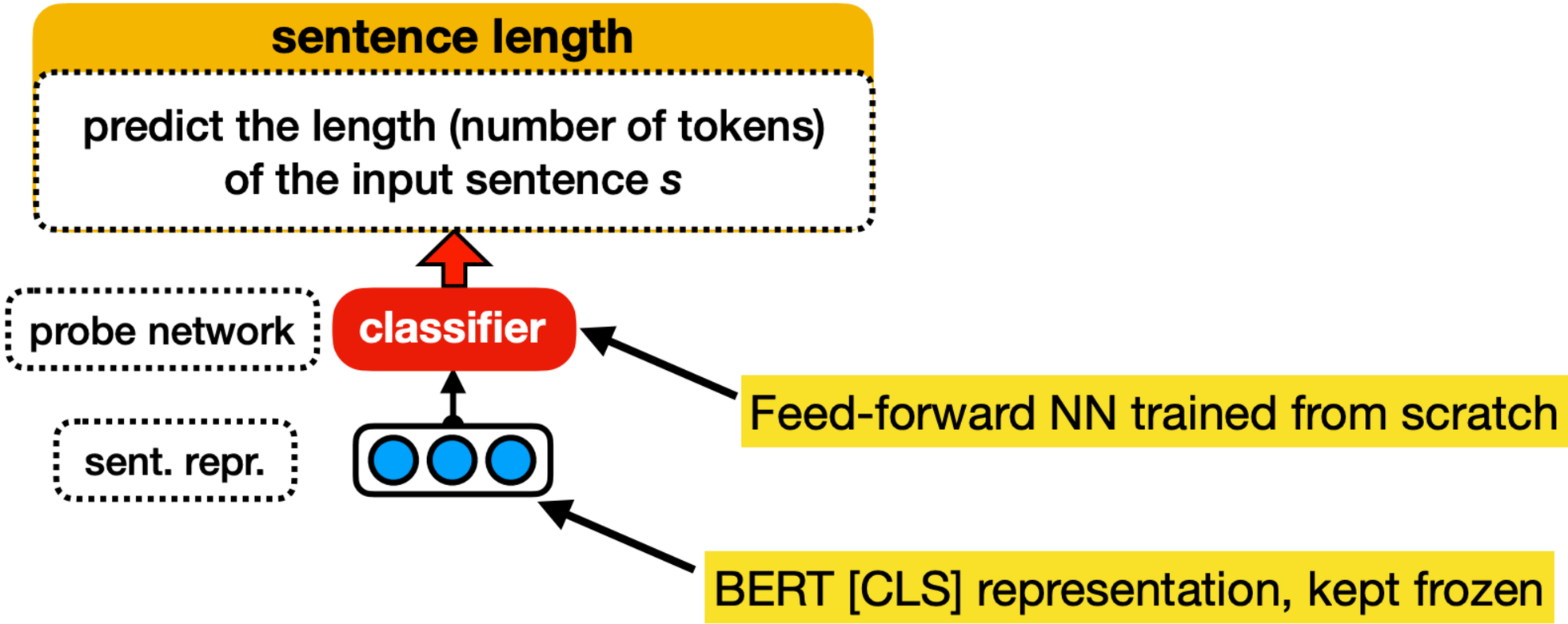
Types of Linguistic Probing Tasks



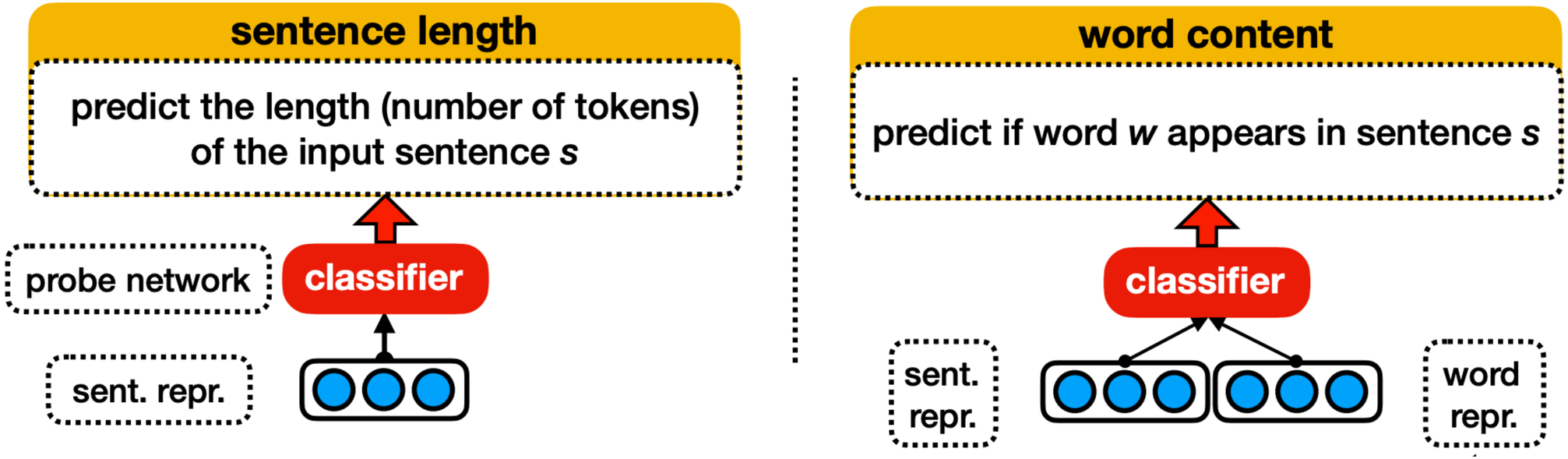
Types of Linguistic Probing Tasks



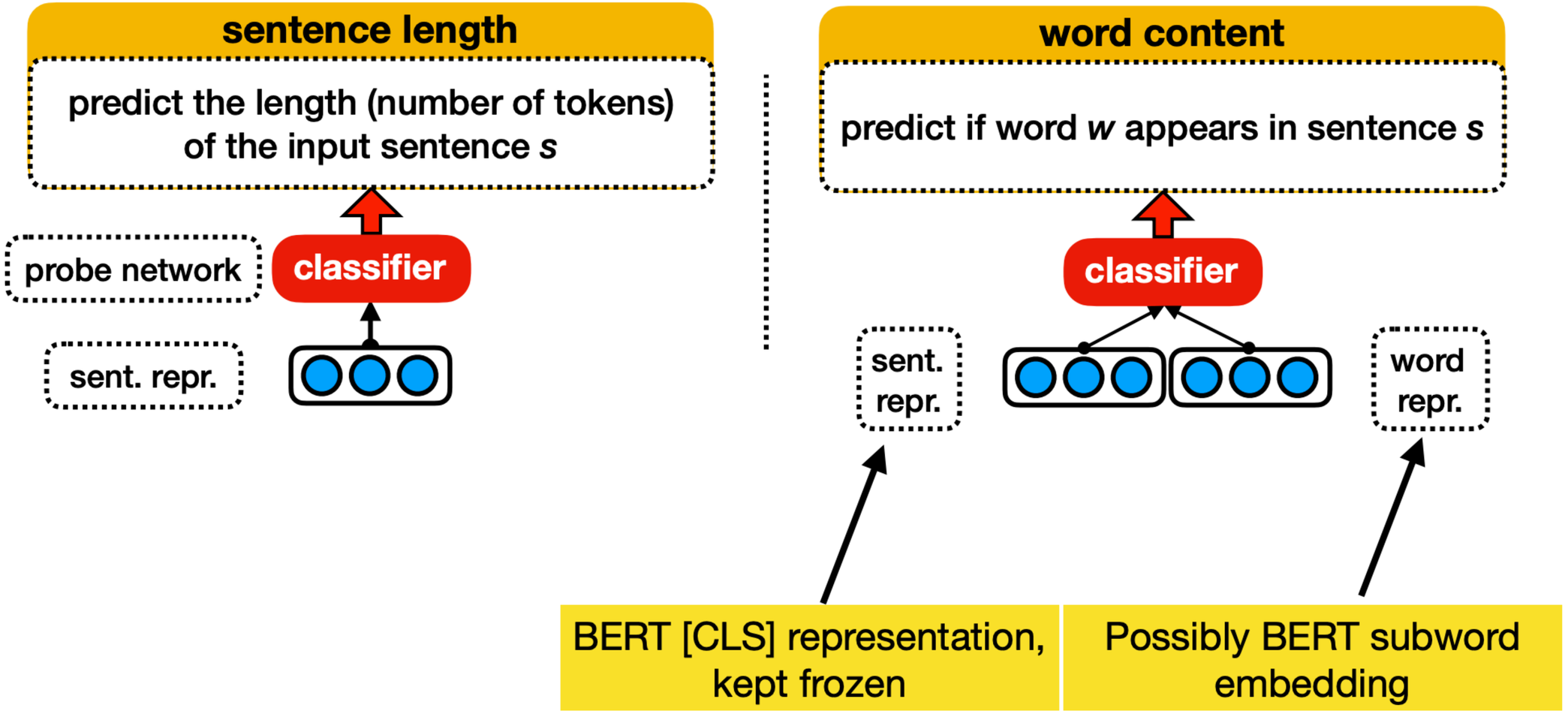
Types of Linguistic Probing Tasks



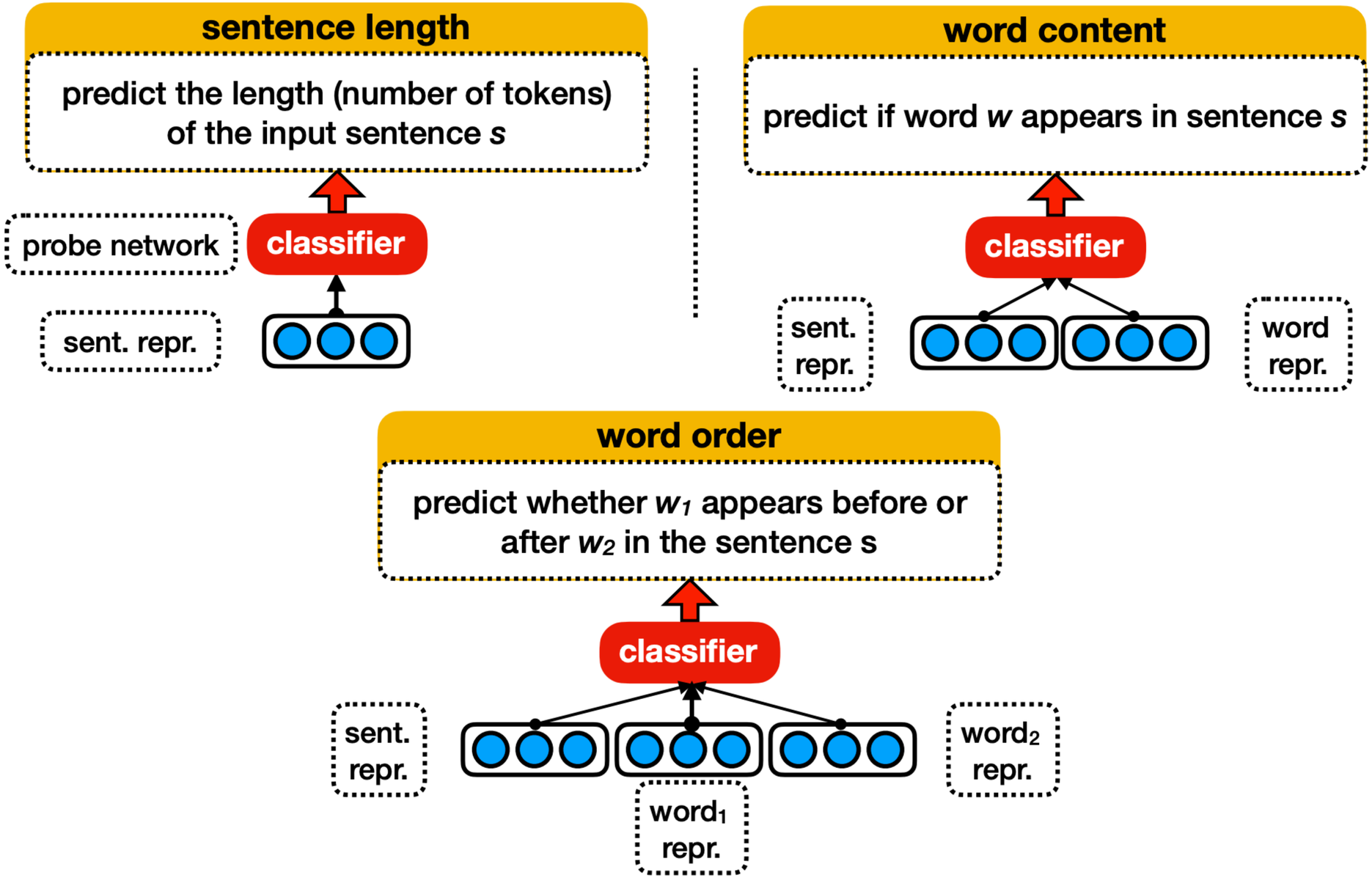
Types of Linguistic Probing Tasks



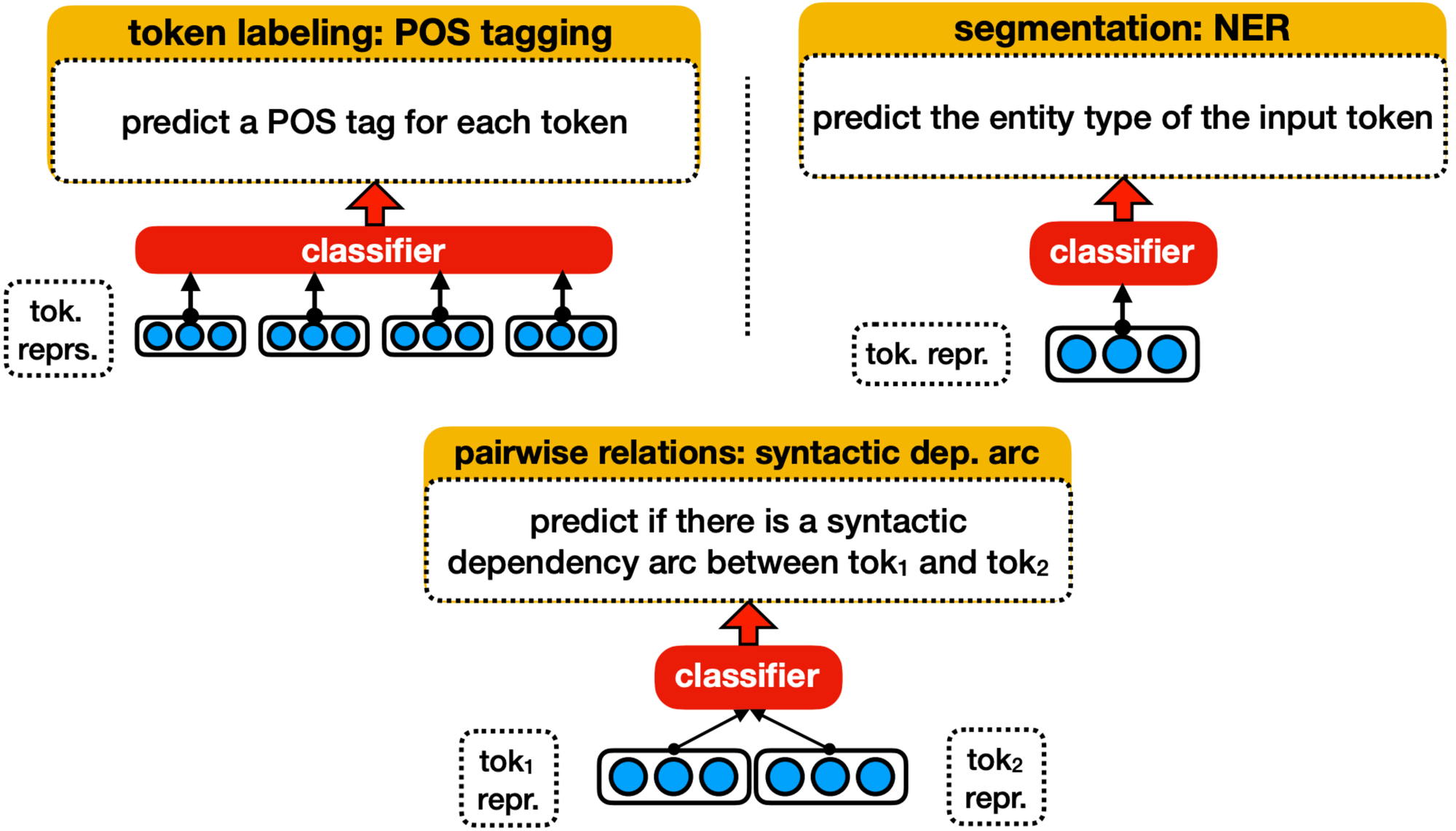
Types of Linguistic Probing Tasks



Types of Linguistic Probing Tasks



Types of Linguistic Probing Tasks



Probing Motivation

- if we can train a classifier to predict a property of the input text based on its representation, it means the property is encoded somewhere in the representation

- if we cannot train a classifier to predict a property of the input text based on its representation, it means the property is not encoded in the representation or not encoded in a useful way, considering how the representation is likely to be used

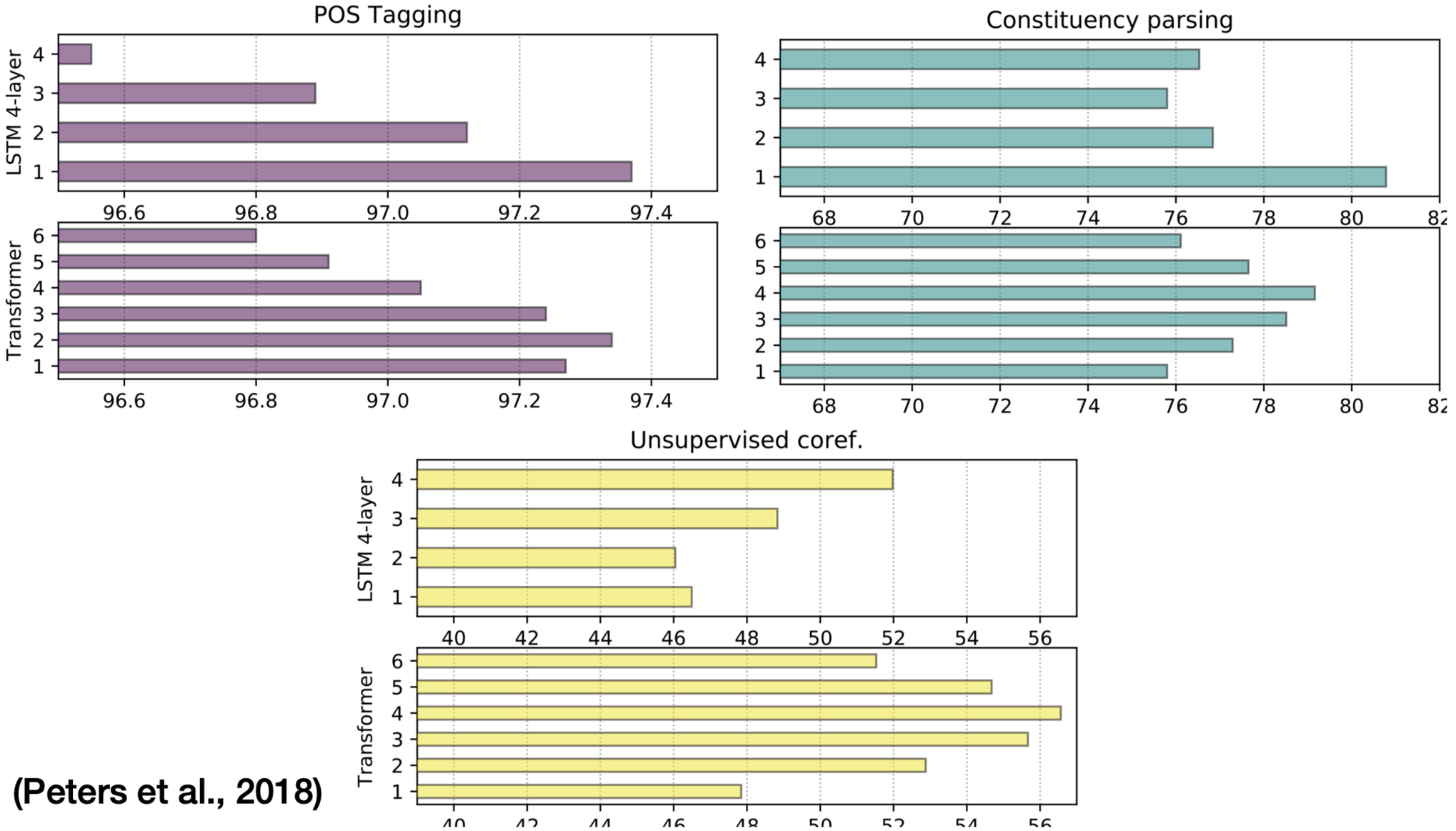
Probing Interpretation

- if we can train a classifier to predict a property of the input text based on its representation, it means the property is encoded somewhere in the representation

- if we cannot train a classifier to predict a property of the input text based on its representation, it means

Nothing?!

Early Layers Capture Local Syntax, Later Layers Focus on Content

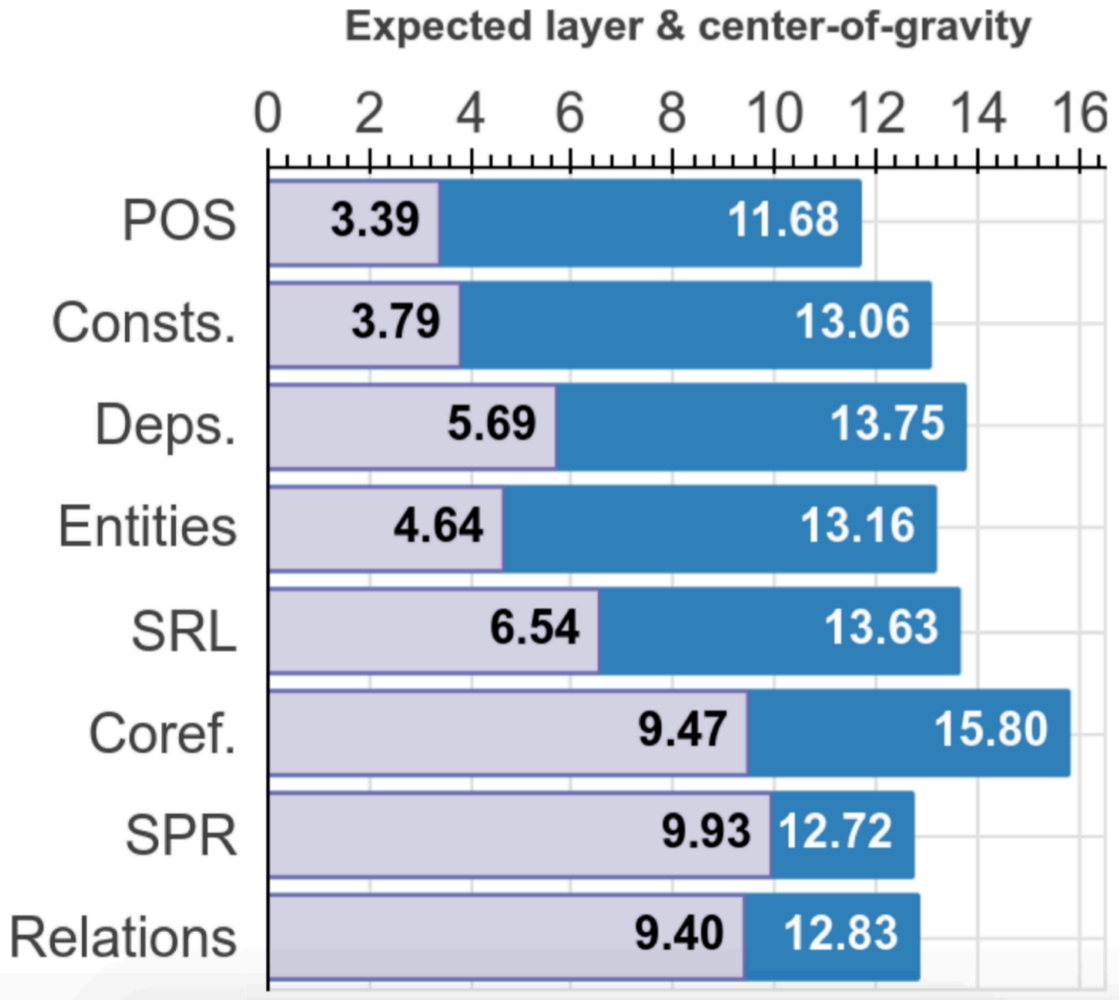


(Peters et al., 2018)

Localizing Task-Specific Computations

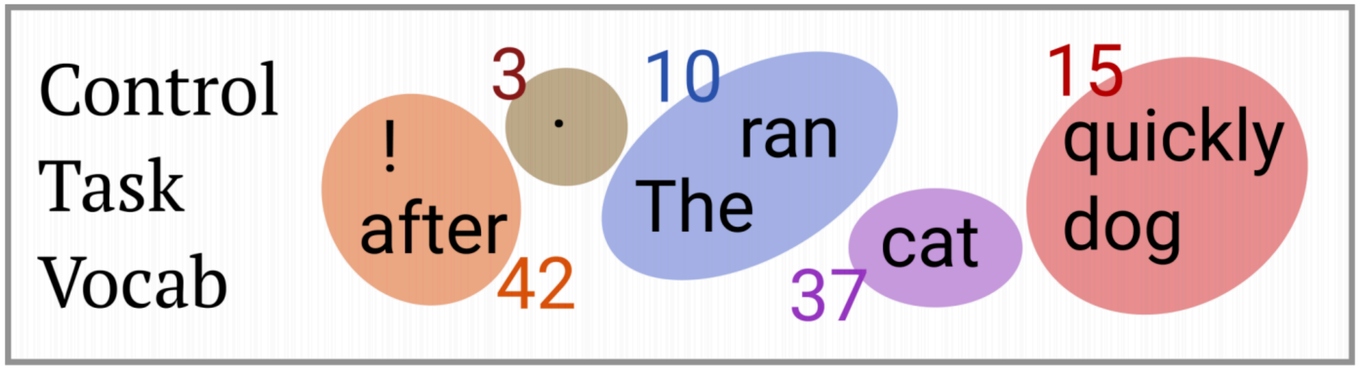
the expected layer at which the probing model correctly labels an example

a higher center-of-gravity means that the information needed for that task is captured by higher layers



Control Tasks [Hewitt et al., 2019]

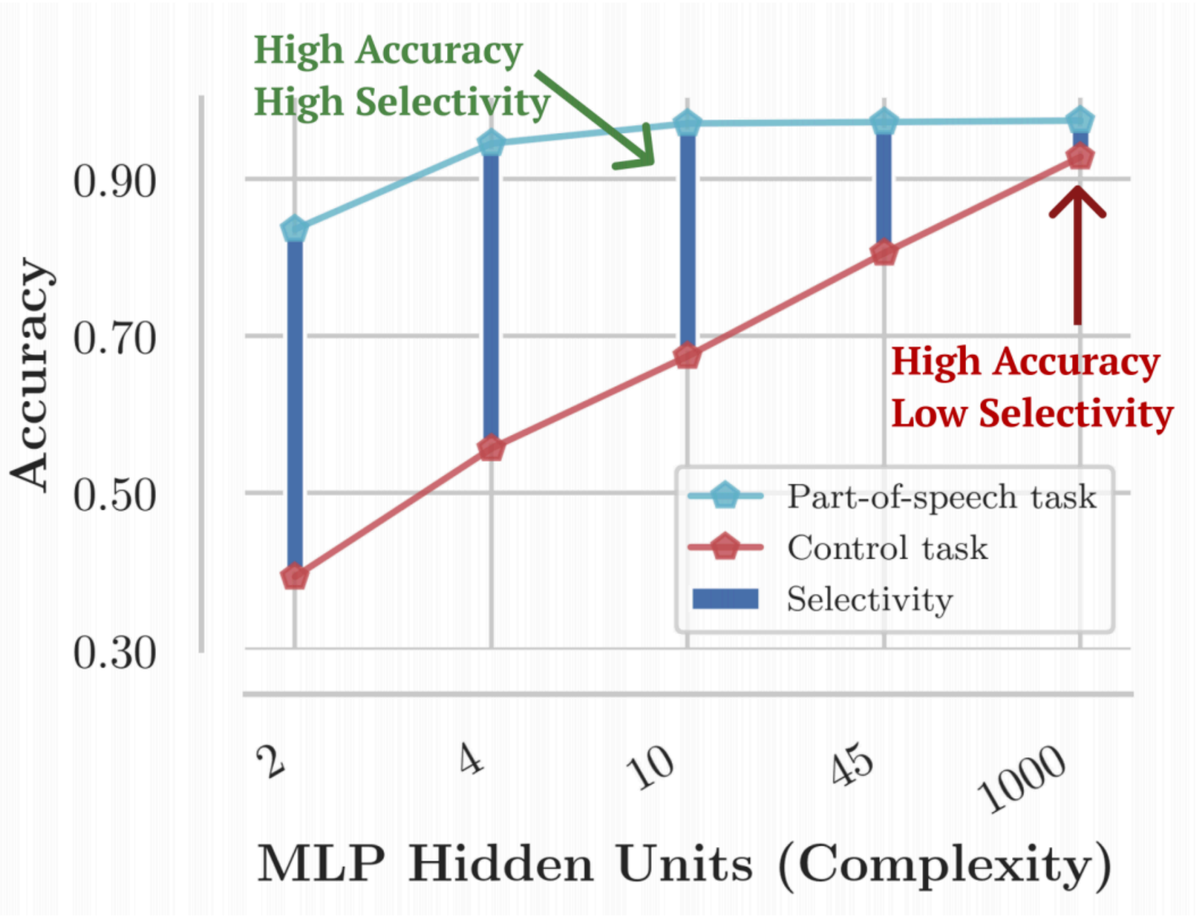
- independently sample a control behavior C(v) for each word type v in the vocabulary
- specifies how to define yi ∈ Y for a word token xi with word type v
- control task is a function that maps each token xi to the label specified by the behavior C(xi)



Sentence 1	The	cat	ran	quickly	.
Part-of-speech	DT	NN	VBD	RB	.
Control task	**10**	**37**	**10**	**15**	**3**
Sentence 2	The	dog	ran	after	!
Part-of-speech	DT	NN	VBD	IN	.
Control task	**10**	**15**	**10**	**42**	**42**

Probe Selectivity

measures the probe model's ability to make output decisions independently of linguistic properties of the representation

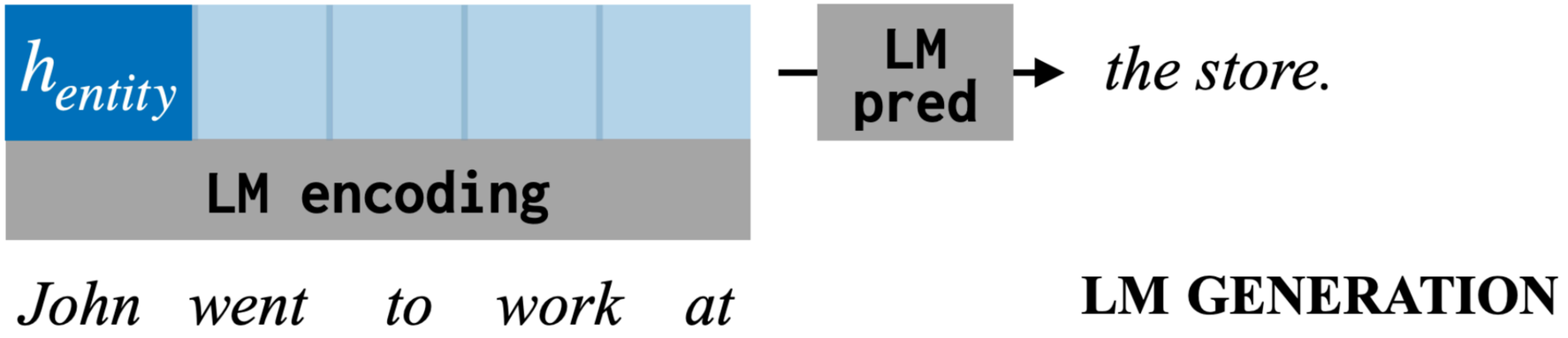


Probing for a Better Tomorrow

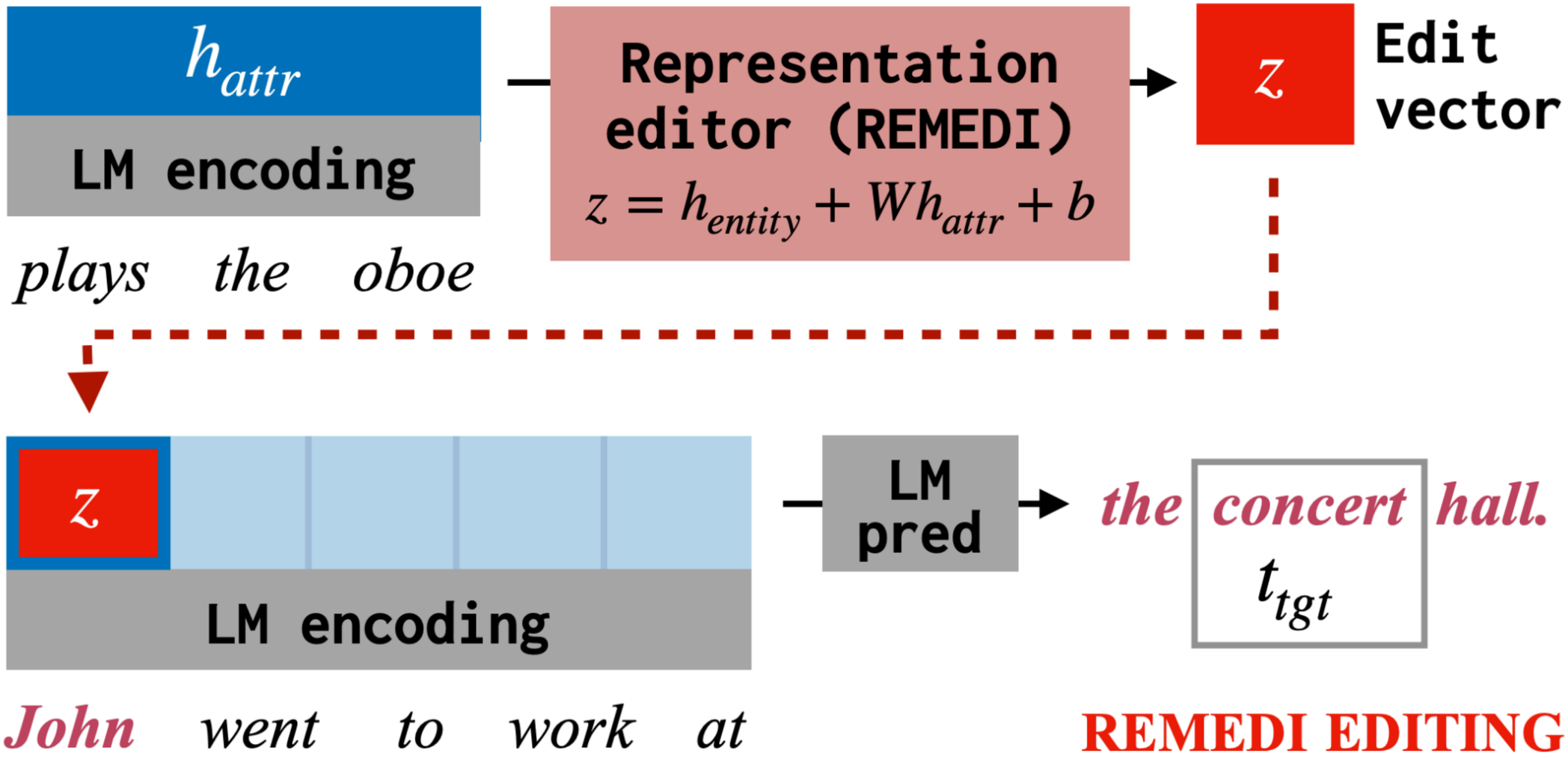
- what kinds of linguistic knowledge are important for your task?
- probe BERT for them
- if BERT struggles then fine-tune it with additional probe objectives

$$\mathcal{L}_{new} = \mathcal{L}_{BERT} + \alpha \mathcal{L}_{probe}$$

Knowledge Editing



Knowledge Editing



Knowledge Editing

Leonhard Euler	domain of activity is opera	✗ **Leonhard Euler is** the most prolific mathematician of the 18th century. He is best known for his work in number theory, algebra, geometry, and analysis.	✓ **Leonhard Euler is** a composer of opera. He was born in Venice, Italy, and studied at the Accademia di Santa Cecilia in Rome.
Microsoft Internet Explorer 6	a product created by Google	✗ **Microsoft Internet Explorer 6 is** a web browser developed by Microsoft for Windows. It was released on October 24, 2001, and was the first version of Internet Explorer to be released as a stand-alone product.	✓ **Microsoft Internet Explorer 6 is** a web browser developed by Google. It is the default web browser on Android.
Beef bourguignon	that was formulated in Canada	✗ **Beef bourguignon is** a French dish of braised beef in red wine, onions, and mushrooms. It is a classic of French cuisine.	✓ **Beef bourguignon is** a Canadian dish. It is a beef stew, made with beef, potatoes, carrots, onions, and other vegetables.

Recap

What just happened!?

- **Best practices for debugging**
- **Fiddling with the inputs**
  - What-Ifs
  - SHAP
- **Staring at weights**
  - Attention Visualization
  - Gradient Tracing
- **Dissecting frozen models**
  - Probing

What is to come…

- Tutorial **June 14th**
- Next lecture **June 18th**
  - Behavioral Assessments