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

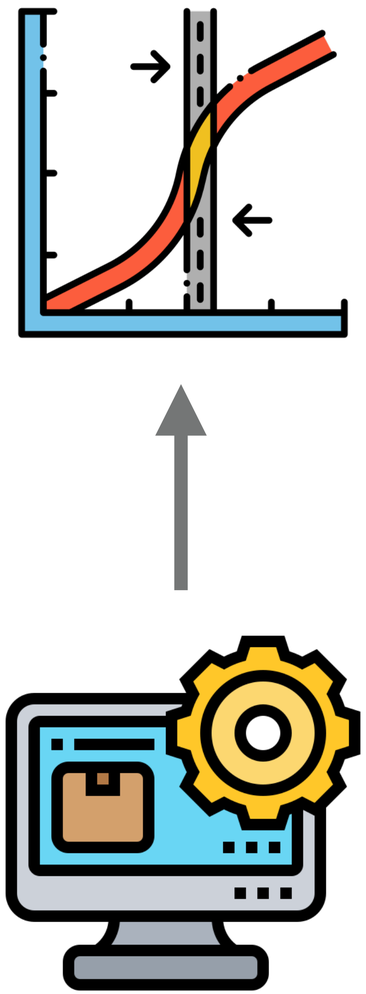
Q: “how asymmetric are the white bricks on either side of the building”
A: *very*

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

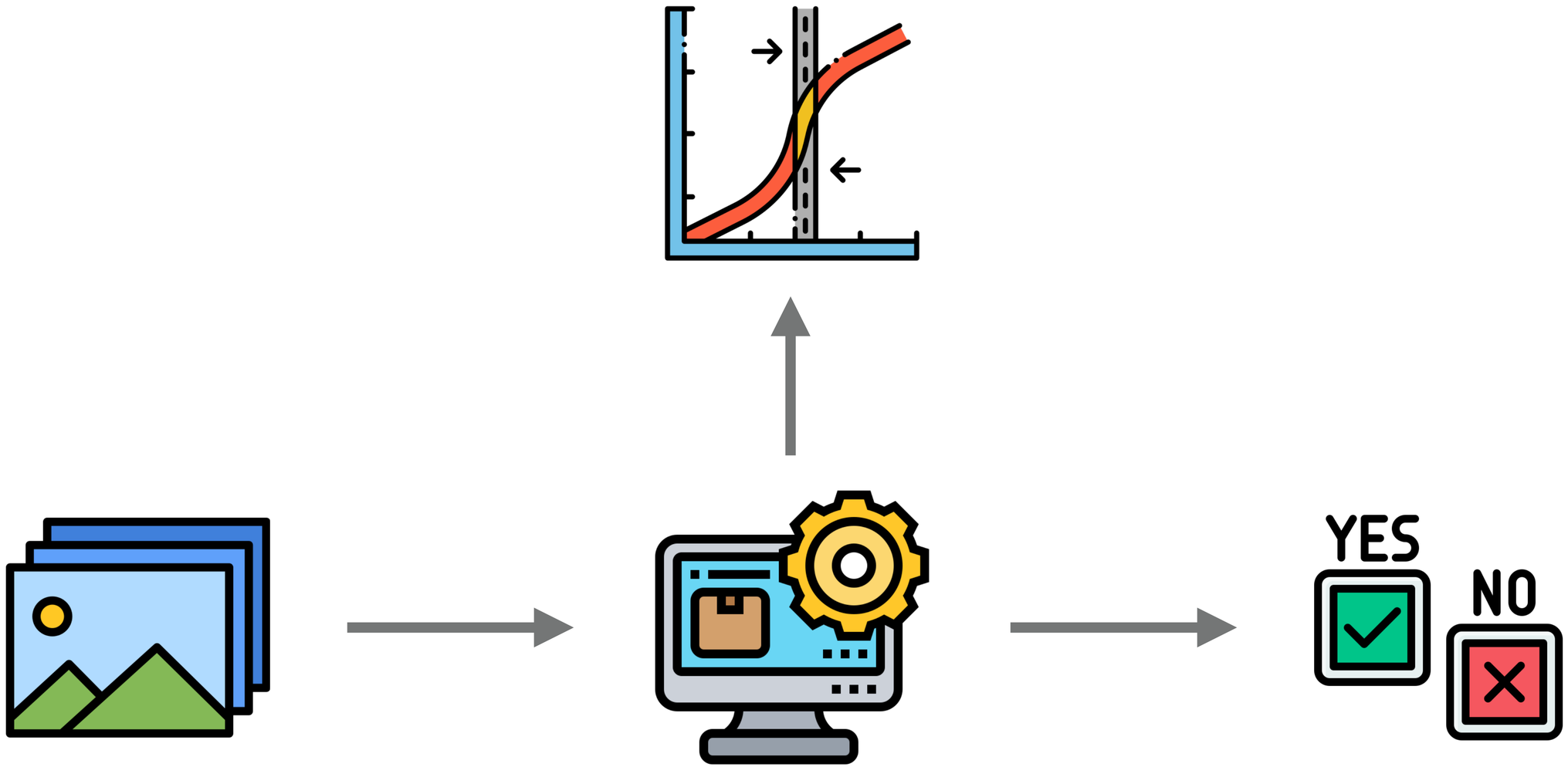
Q: “how fast are the bricks speaking on either side of the building”
A: *very*

Paper: Did the model understand the question? ACL 2018

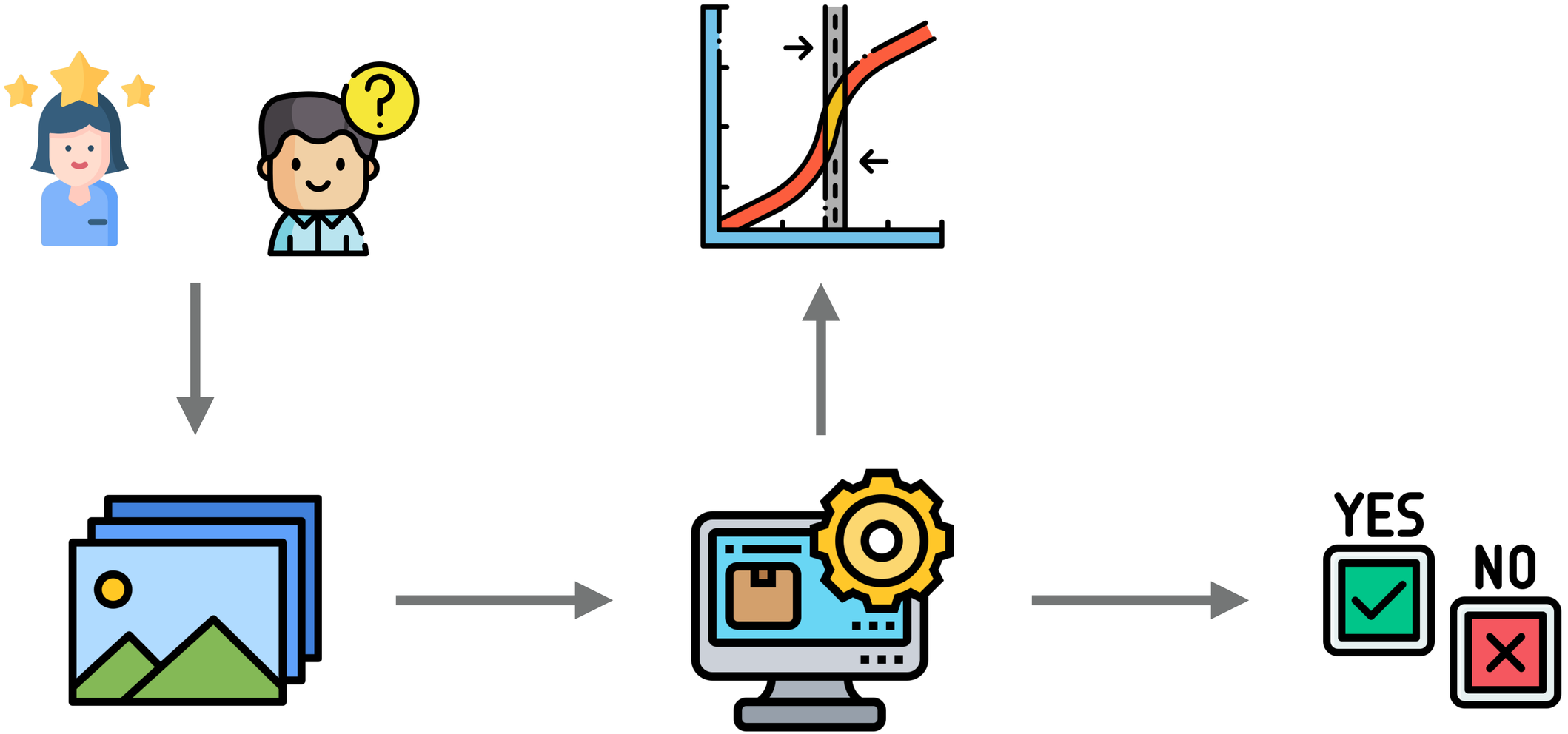
Global Explanations



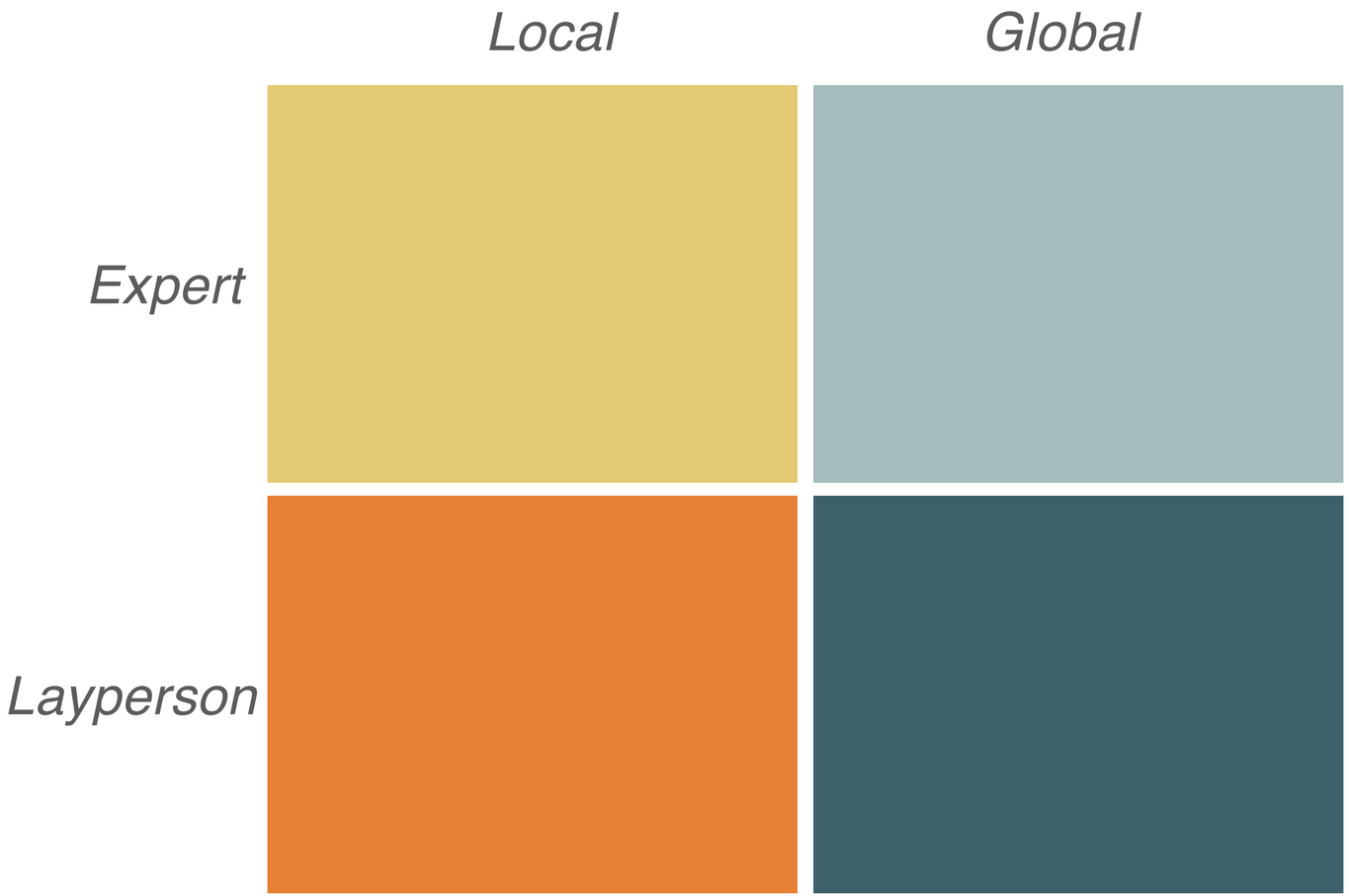
Local Explanations



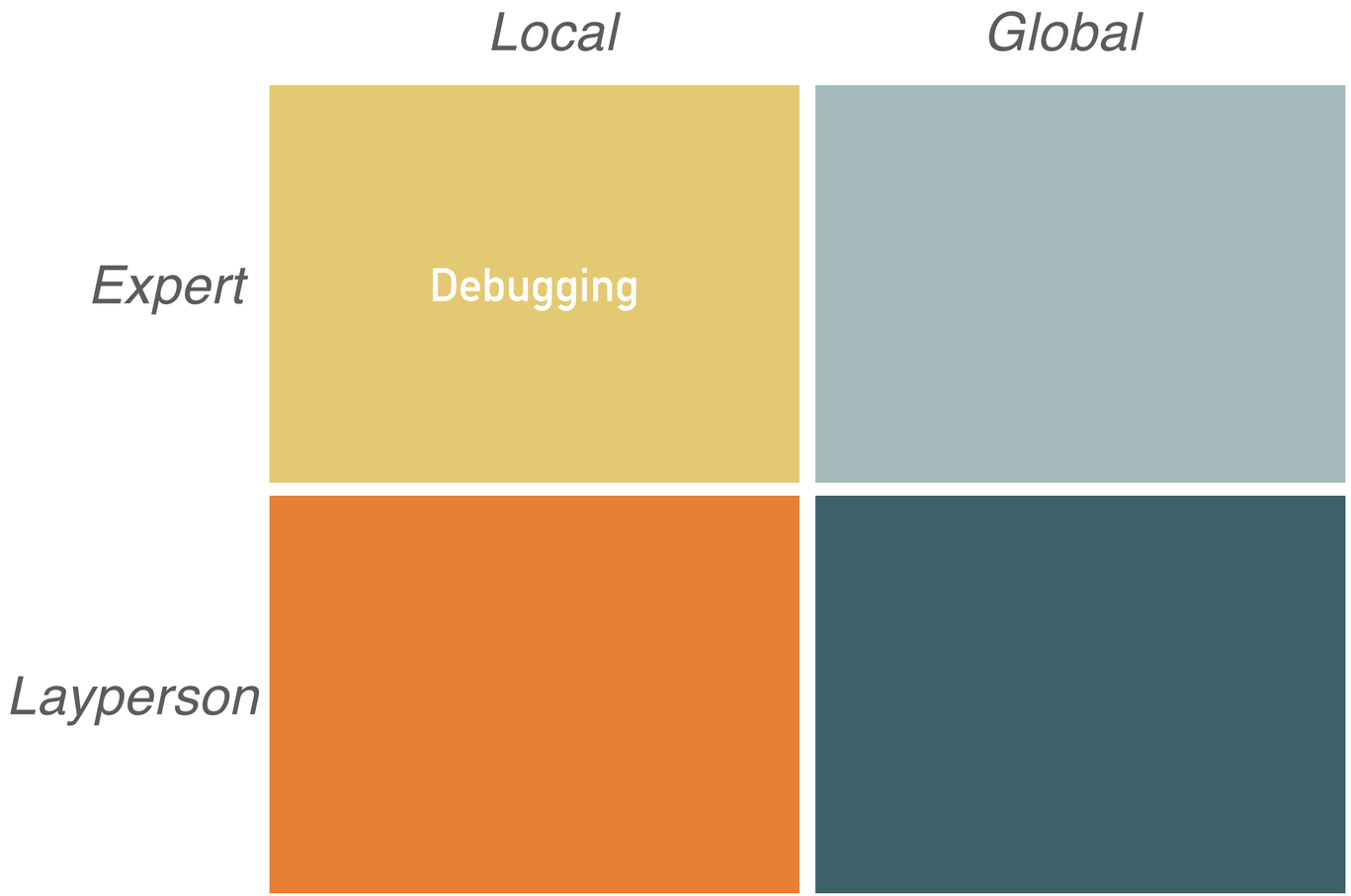
Suitable Explanations



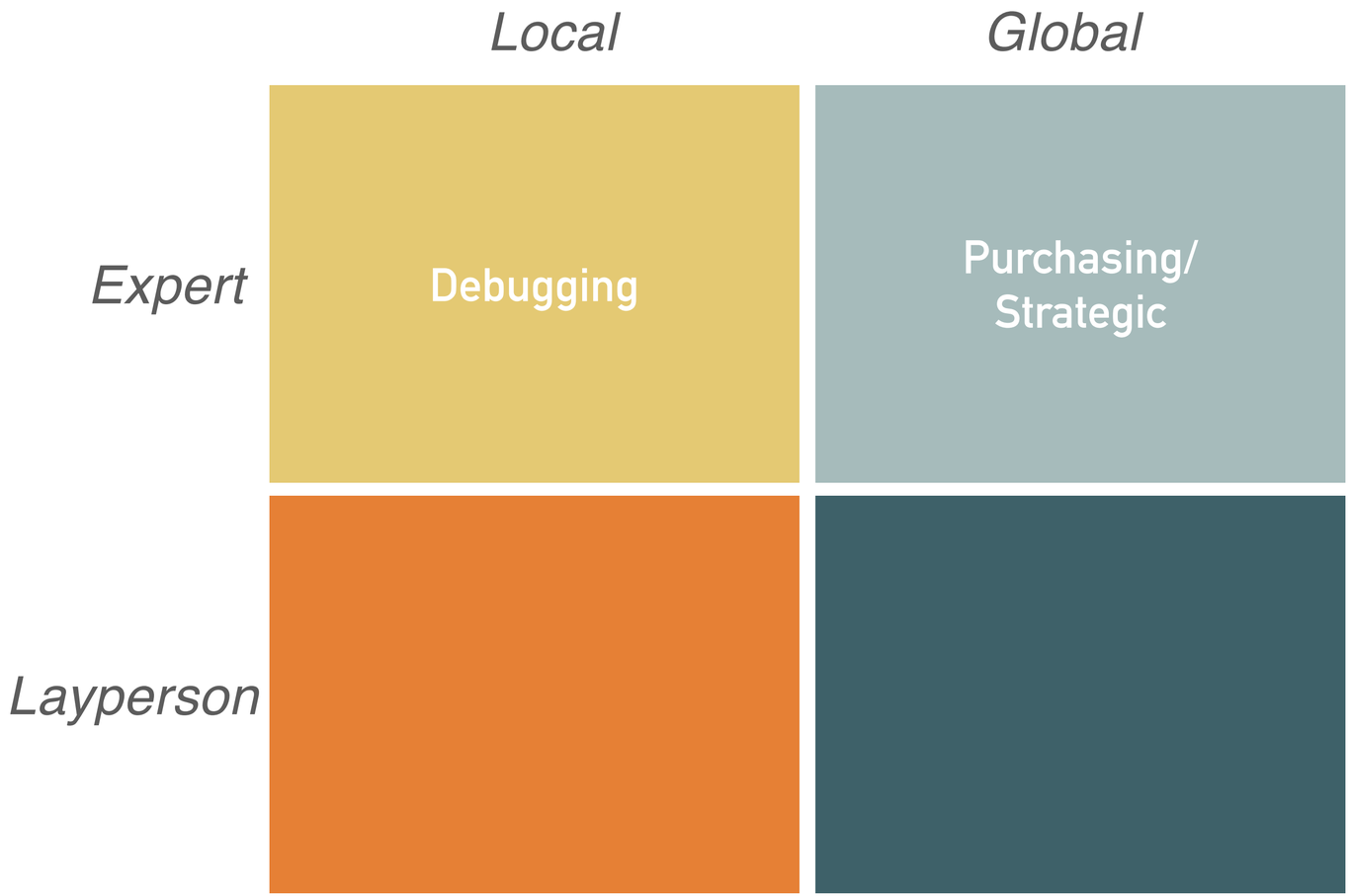
Tailored Explanations



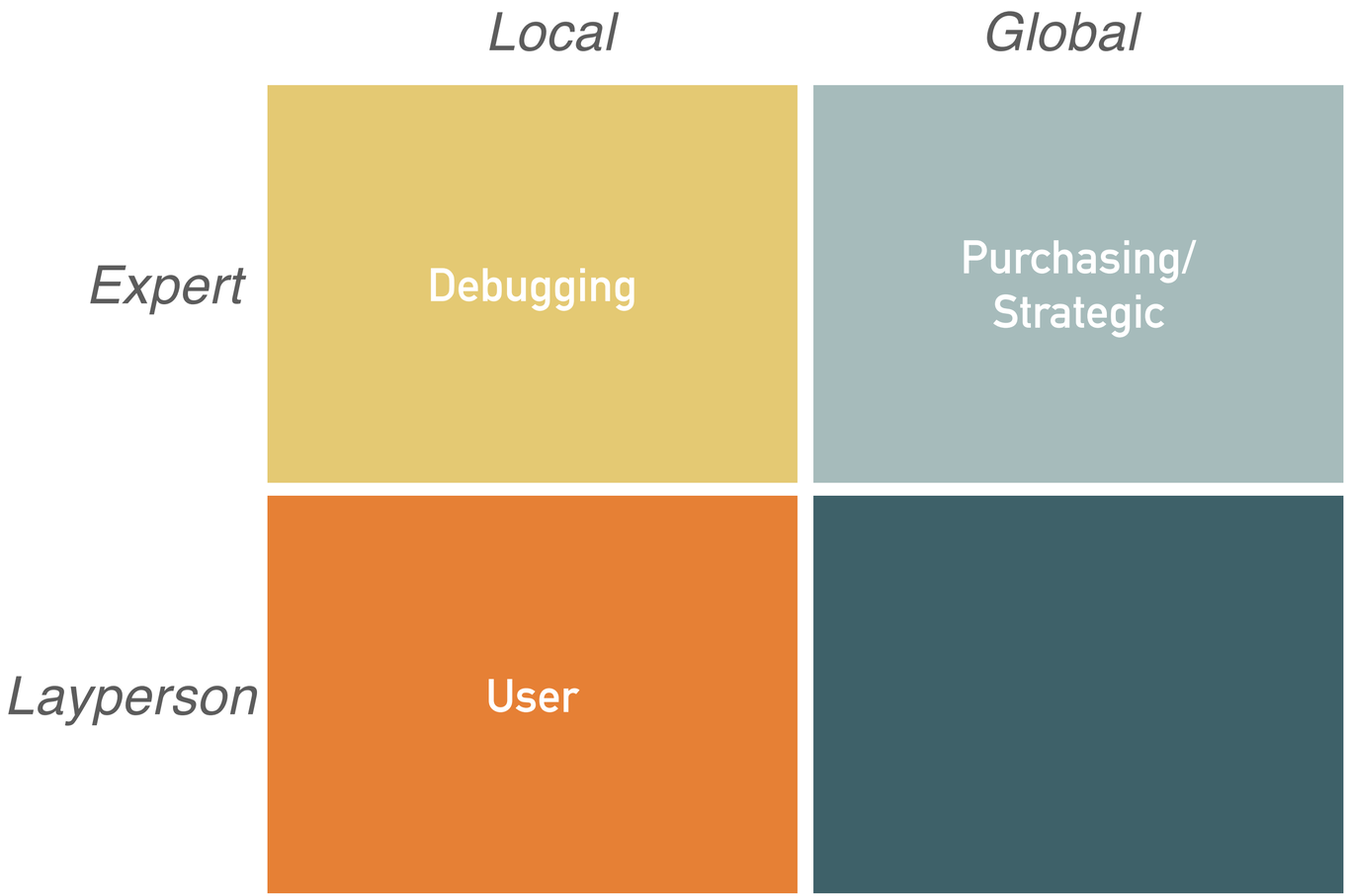
Tailored Explanations



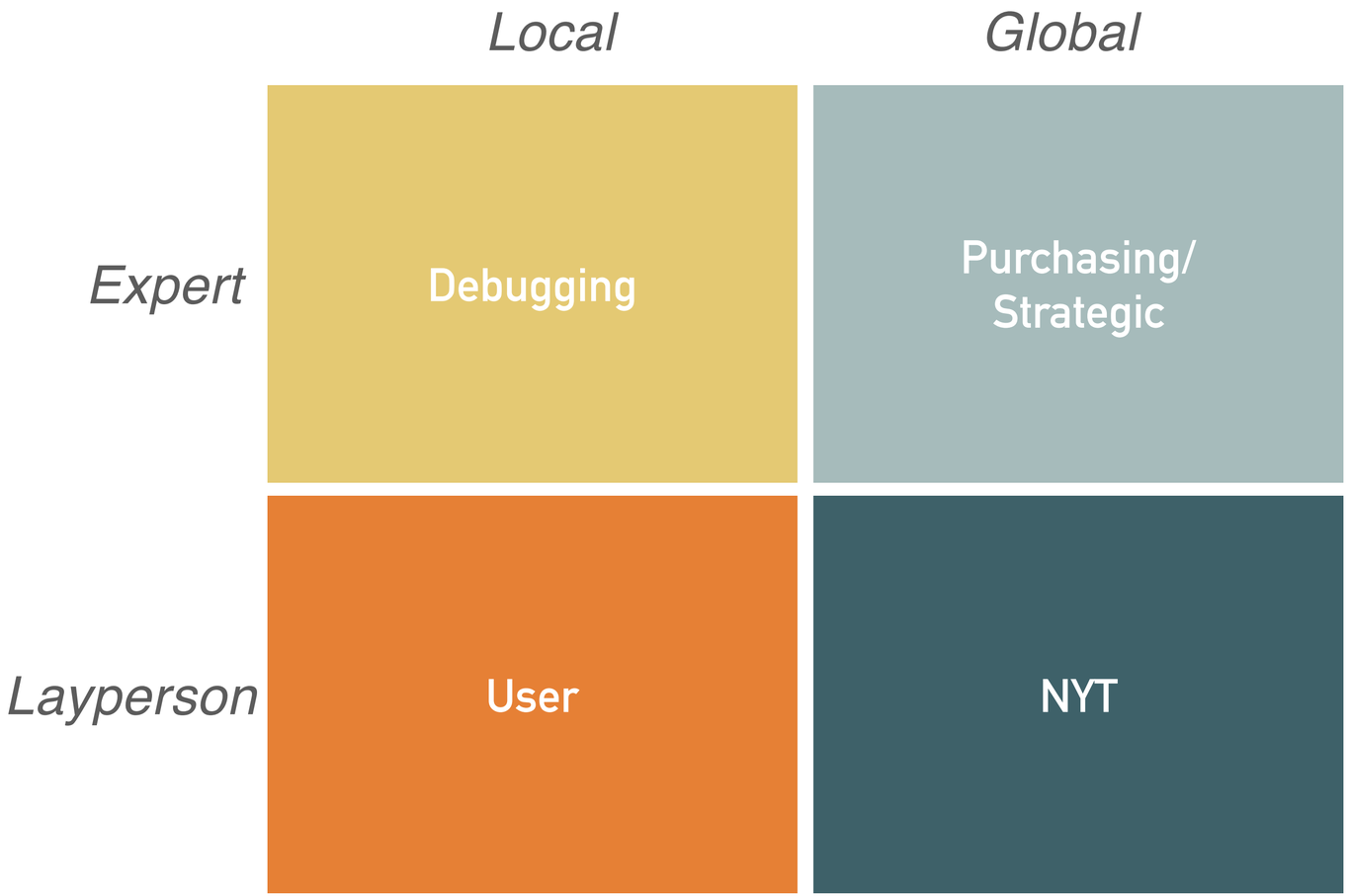
Tailored Explanations



Tailored Explanations



Tailored Explanations



What-if Explorations

the process of using the model to explore hypothetical scenarios

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

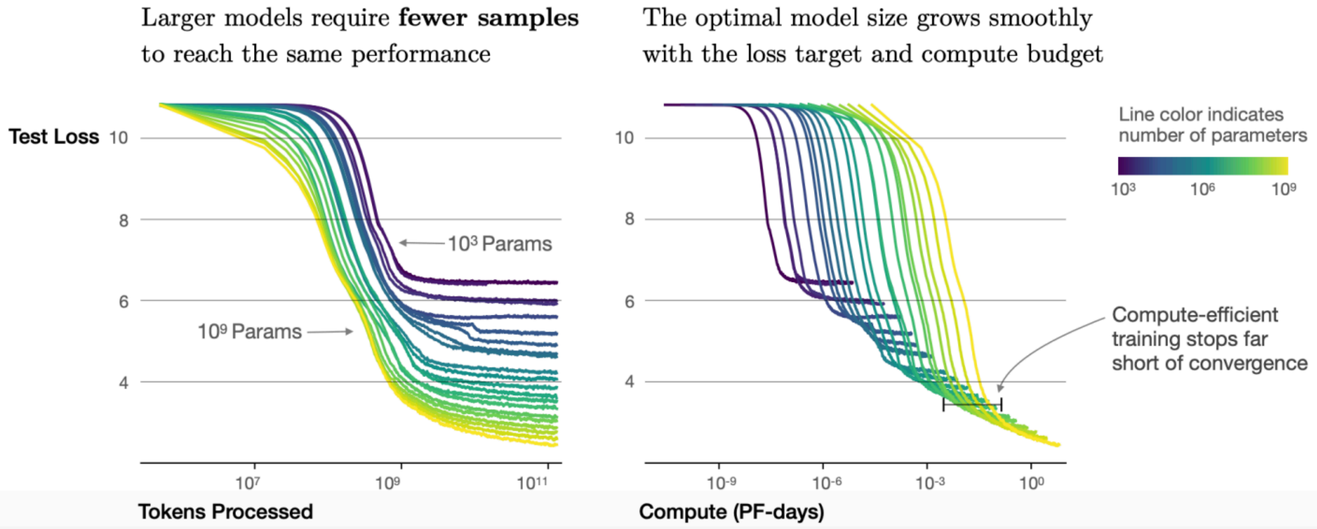
- Look at the **loss function** calculated on the **training set**
  - Is the loss function going down?
  - Is it going down basically to zero if you run training long enough (e.g. 20-30 epochs)?
  - If not, does it go down to zero if you use very small datasets?

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

but small difference

bigger influence,
e.g learning step size

did u initialise variables wisely

more, better,
more robust/stable gradient updates

caution: documentation

Training/Test Disconnects

another issue: training & test settings are disconnected

- Usually your loss calculation and prediction will be implemented in different functions properties might vary
- 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

it's better to:

  - Calculate loss with **large batch size** (e.g. 32)

larger

  - 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

another thing when look at structure generation
such as encdec

- 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!

decoder function (output) = loss function (output)

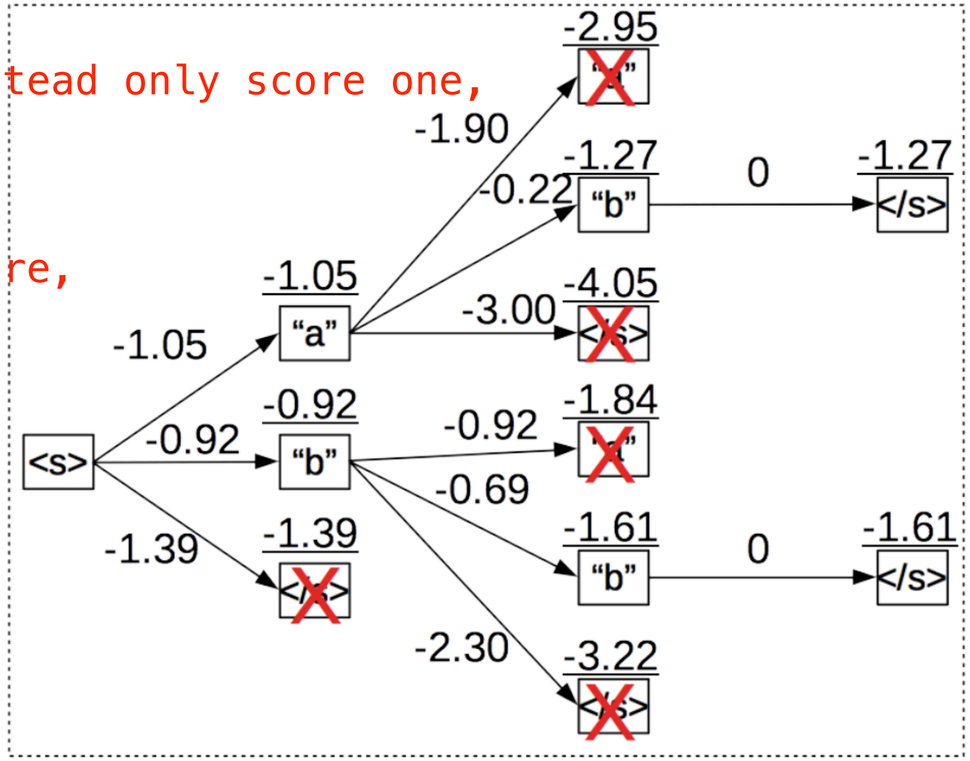
Beam Search

beam search also make things messy

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

have additional promising ones instead only score one,
get the prob, drop the low ones,
end up with a pool of candidates

we expect to have better model score,
if we have broader tree.



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

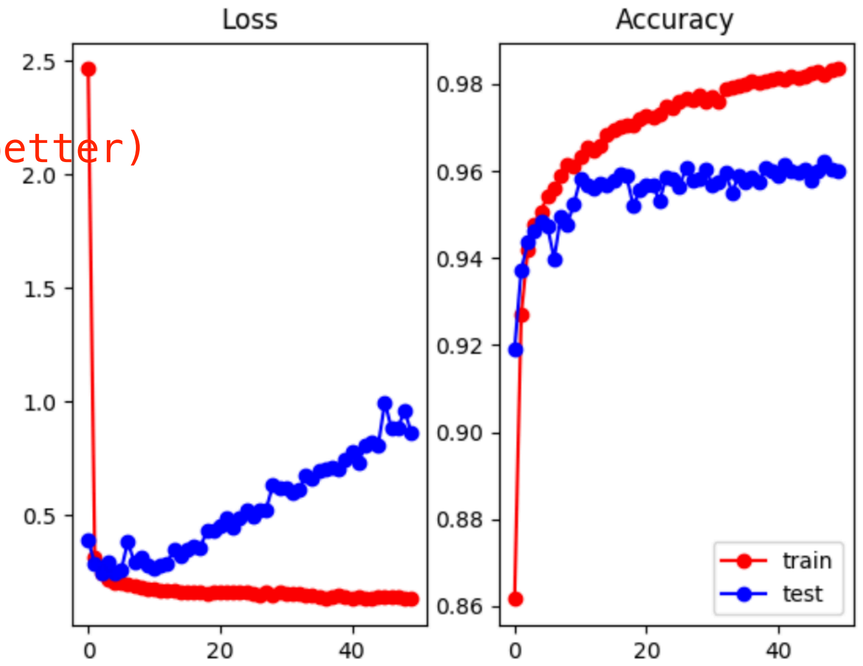
- It is very common to optimize for maximum likelihood for training
- But even though likelihood is getting better, accuracy can get worse

i.e. loss is going down, but accuracy stays the same/worse

Example: Classification

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

See test performance in blue line.
model gets worse (loss goes up),
but accuracy better (performance better)

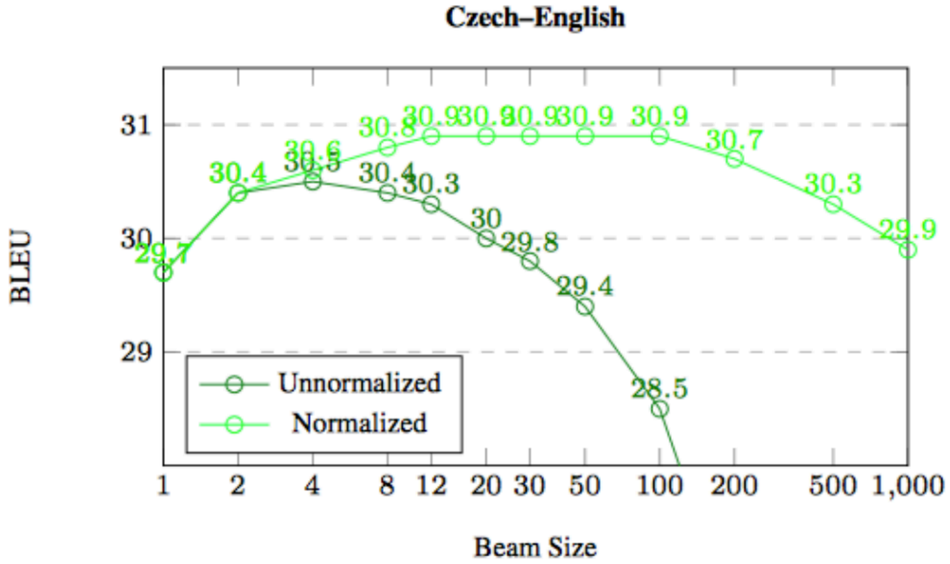


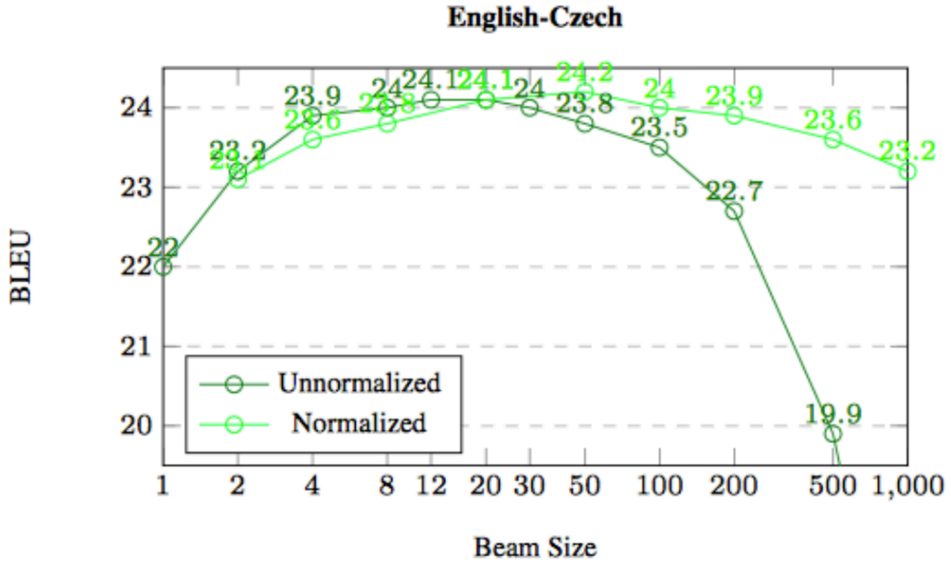
- Why? Model gets more confident about its mistakes.

does my model learn sth that actually make a difference,
or it's stuck in its own way

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.

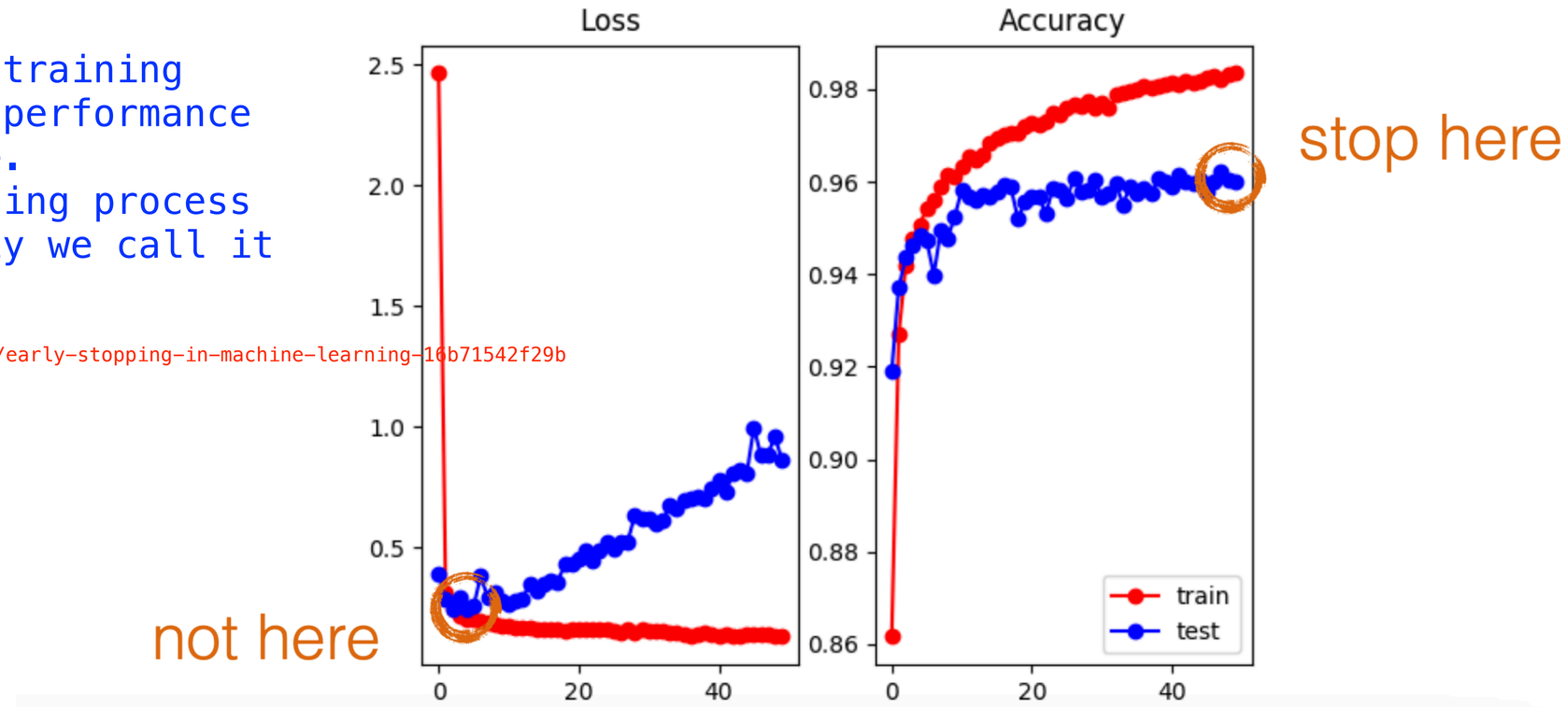
i.e it prefers shorter sentences

Managing Loss/Metric Differences

- Most principled way: use a method like reinforcement learning (also most difficult way)
- Easier way: Early stopping w/ evaluation metric

@OUT: Stop the training process if the performance doesn't improve.
Since the training process is stopped early we call it Early Stopping

https://aoishidas28.medium.com/early-stopping-in-machine-learning-16b71542f29b

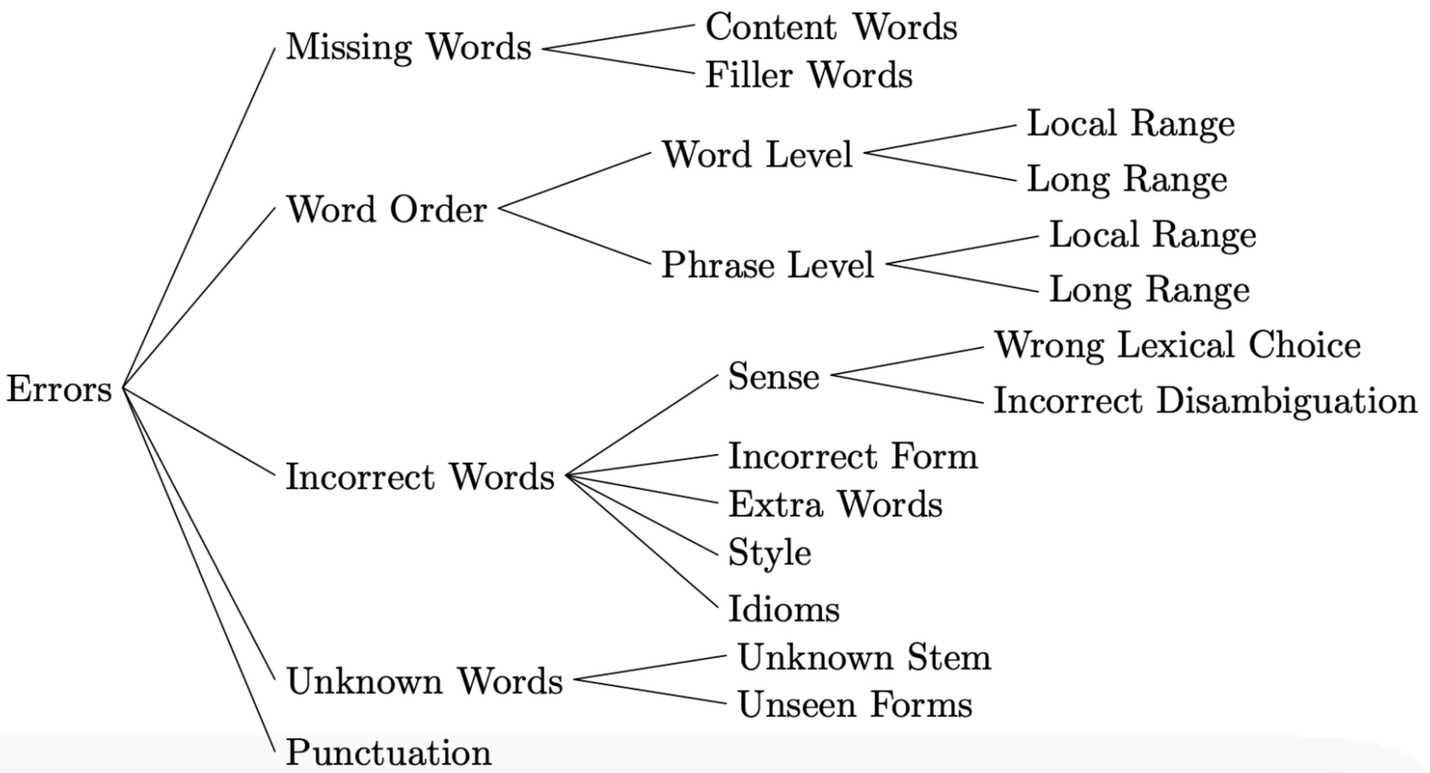


Look at Your Data!

- 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

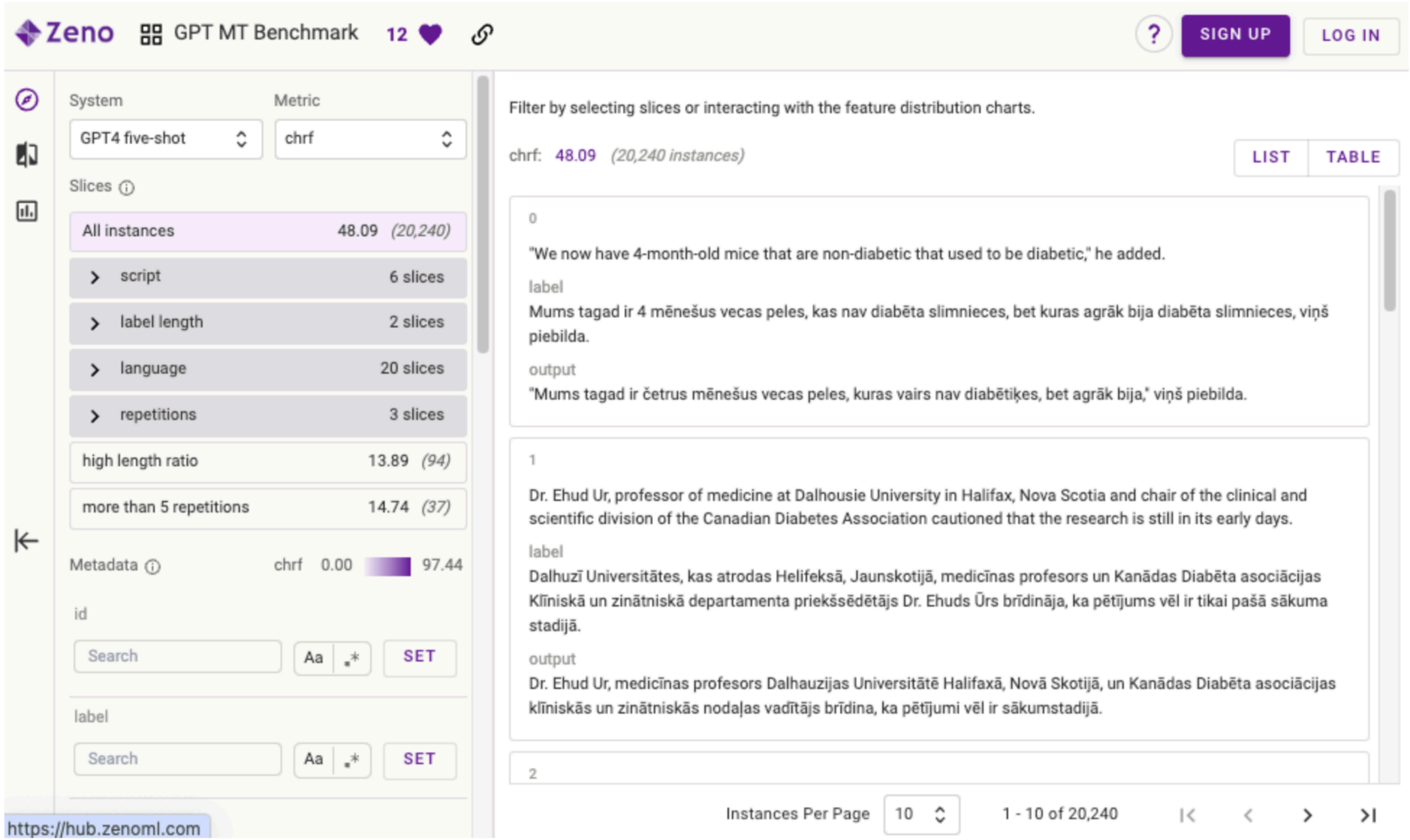
- **Look at 100-200 errors**
- Try to **group them** into a typology (pre-defined or on the fly)
- Example: Vilar et al. (2006)



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

a concrete technique

Shapley Values

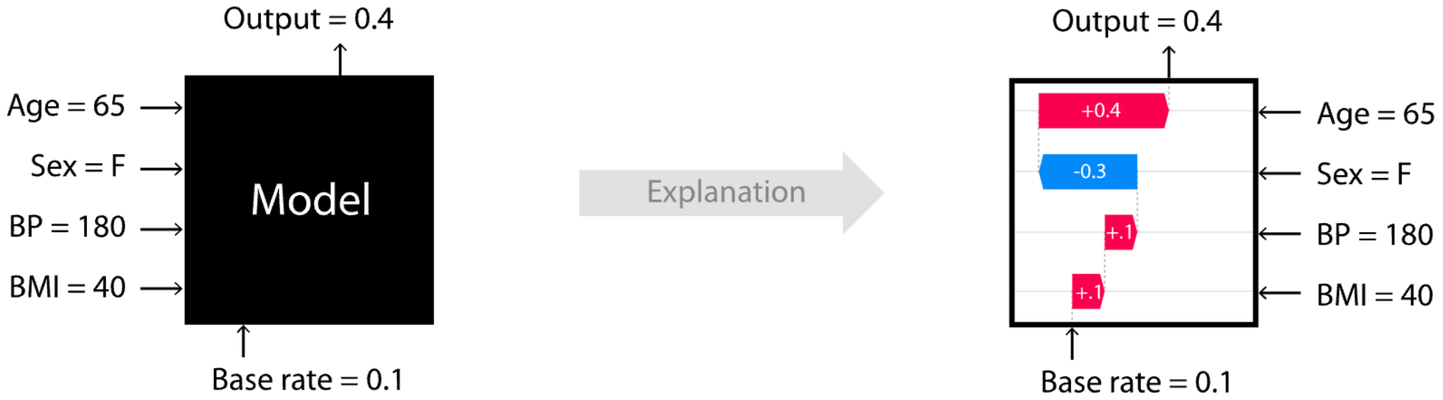
intuition, formalism, and an example

Shapley Values

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

combination, in order

SHAP



idea:
how much/in which direction
do individual features influence the model decision

i.e. what pushes my model towards class 1, 2 … (in classification task)

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

“收益”, 在nlp context是model improvement

- 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 ⊆ {1, …, 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 in proportion

Formal Notion

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

subset of players

- 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

HOW-TO
formulate the coalition,
evaluate how good are the players,in order
particularly, track the marginal contribution each

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]

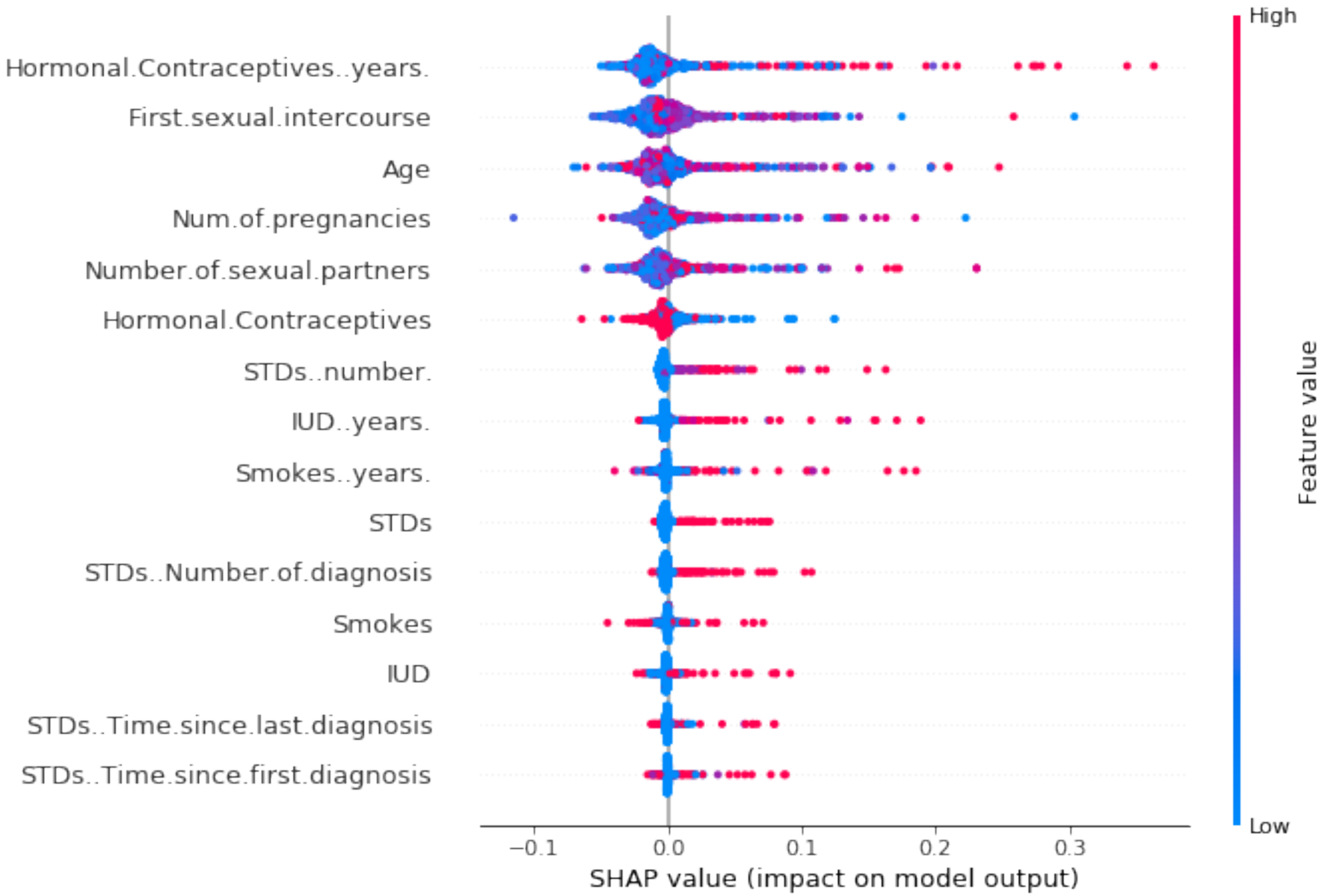
What should the bonuses be?

Permutation	**Marginal for Alice**	**Marginal for Bob**
Alice, Bob	20	30 50−20
Bob, Alice	40 50−10	10
Shapley Value	**30**	**20**

/2

Going Global

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



SHAP for NLP

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

SHAP values indicate whether a token has a positive or negative impact on the prediction.

A positive SHAP value means the token contributes positively to the prediction,
while a negative SHAP value means it contributes negatively.

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

direction: neg or pos

– adapt to any kinds of ML model:
linear regression, decision tree, nn, etc.
– not rely on particular params or structures

computation is complicated. 阶乘

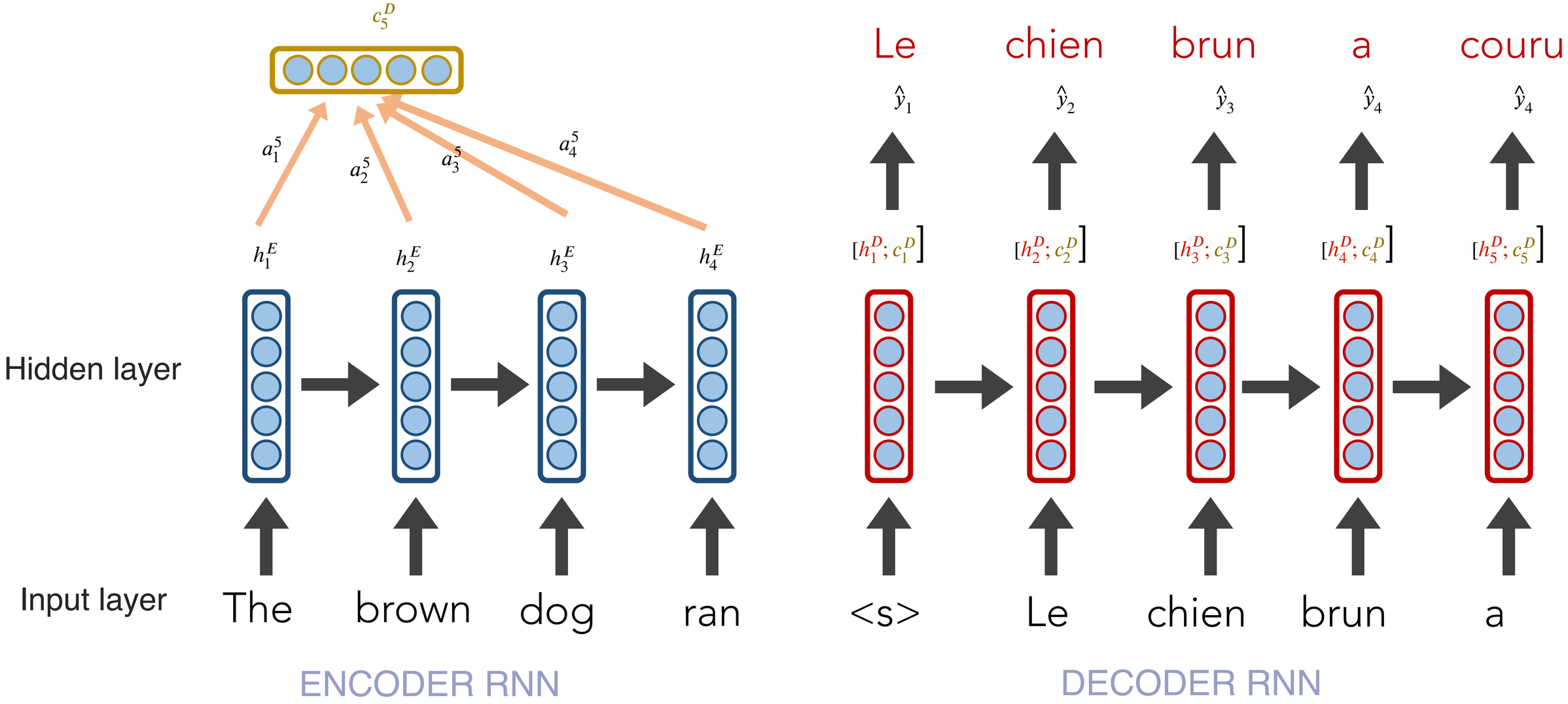
no implicit explanation on model’s underlying mechanism.
i.e. black box

Attention Visualization

another tool/vehicle to debug

Attention Recap

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



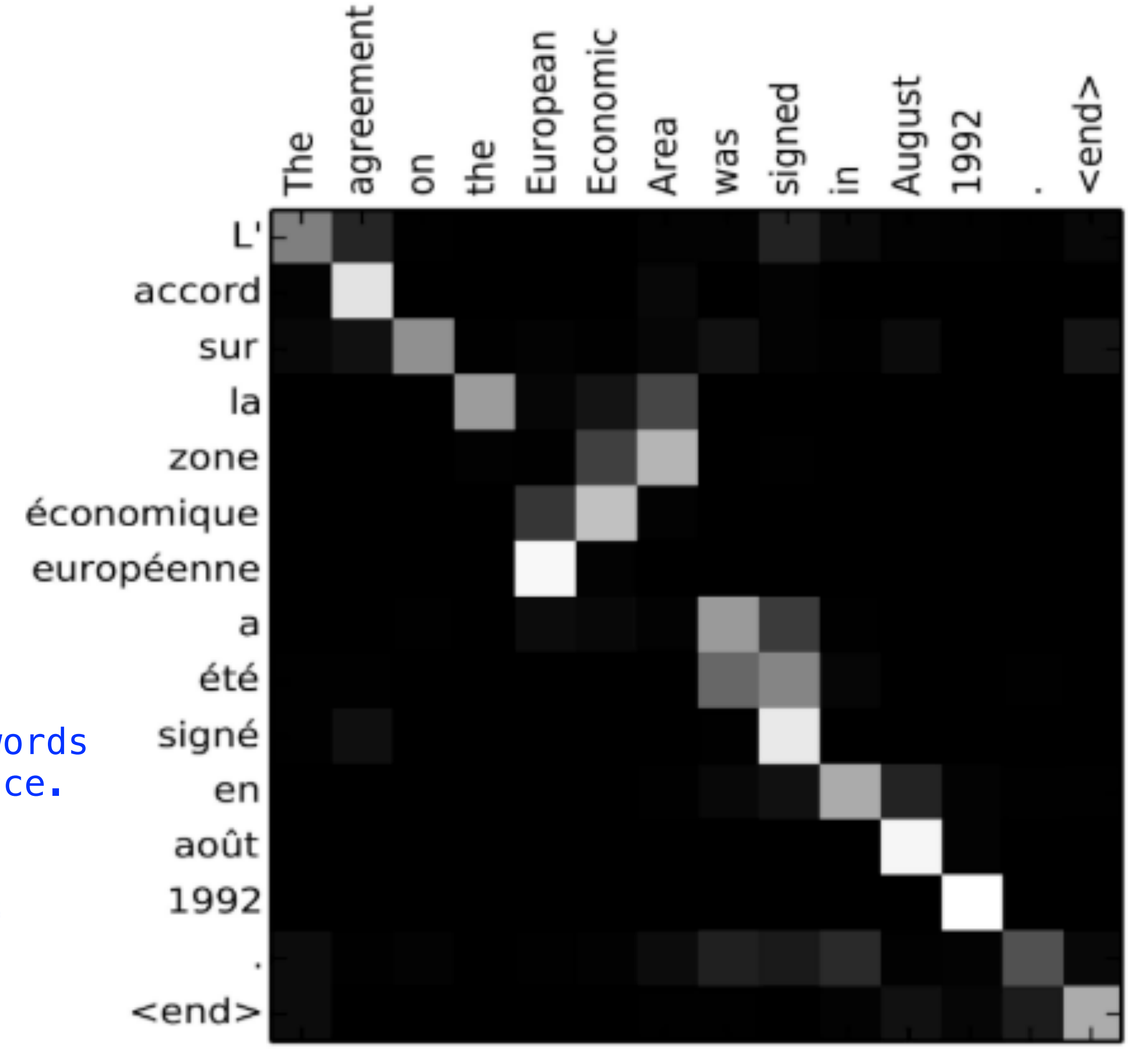
Attention Recap

whiter color are stronger attention

IDEA: How model internally pay attention to these different data points

Attention:

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

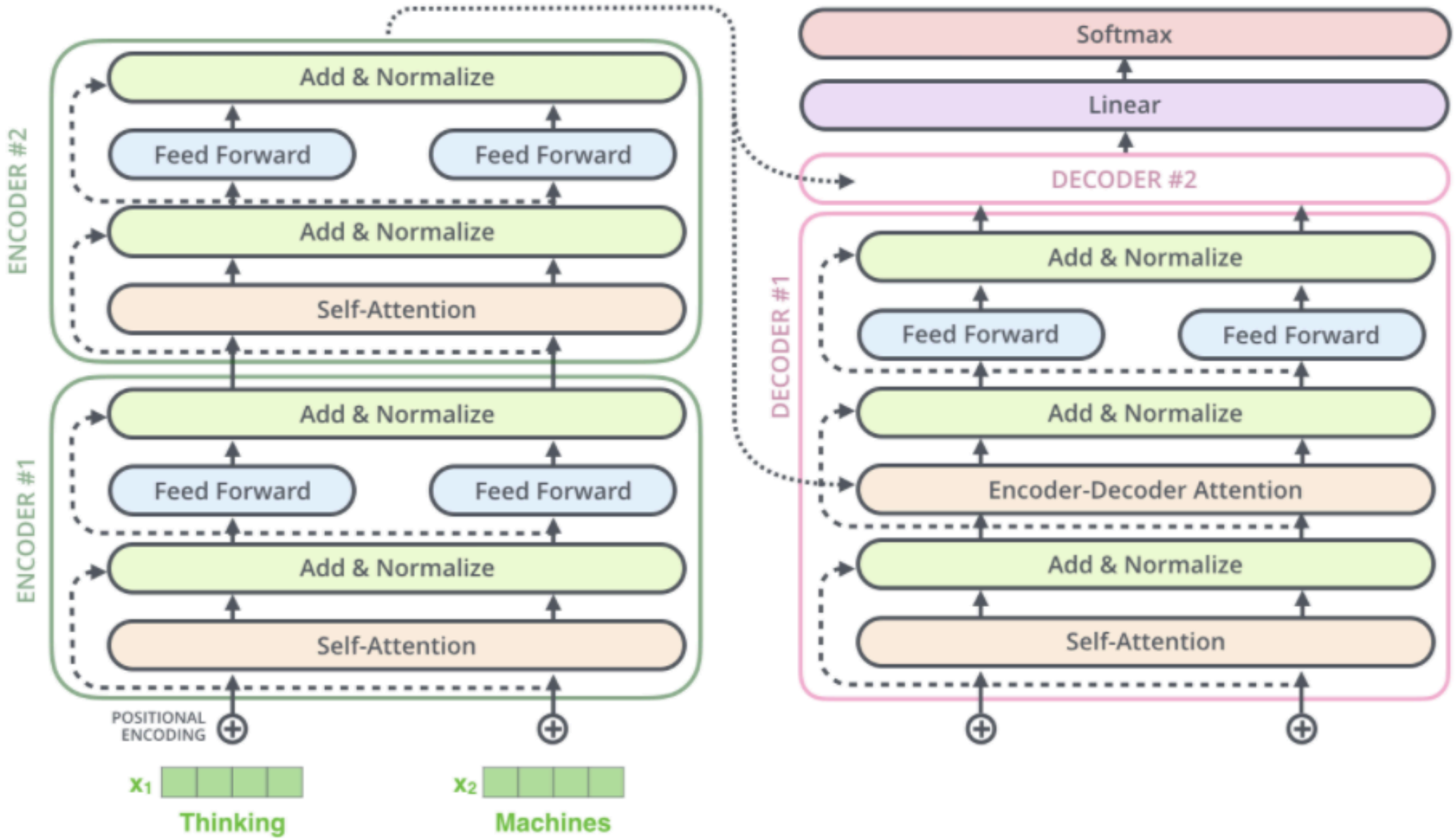


translate from FR to EN

@SOURCE: "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?

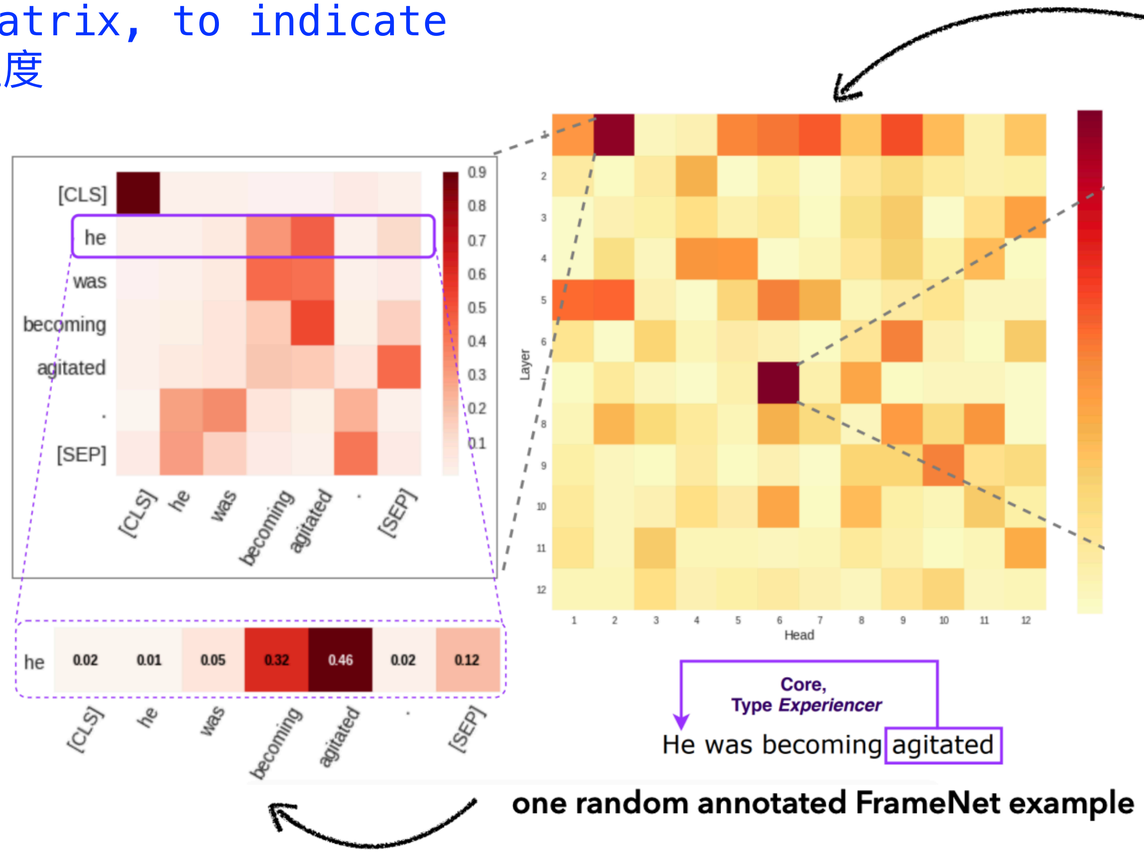


“Interpretable” Attention Heads

identify the attention head related to the problem

– Relation-specific heads

attention weight matrix, to indicate
每个词对其他词的关注程度



HEAD IMPORTANCE

average max absolute attention weight
among FrameNet-annotated tokens

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

Michel et al, 2019:

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

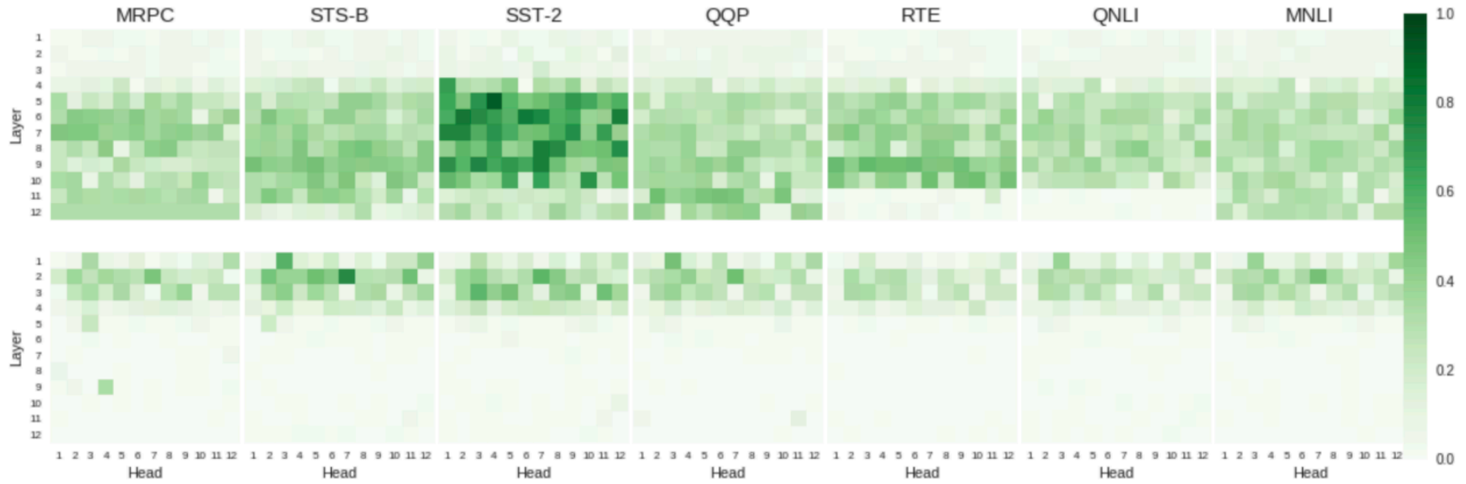
one random annotated FrameNet example

abnormal e.g.:
如果某个词对无关词
有异常高的attention weights

“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

It's controversial.
So Shap & gradient-based is usually better

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

消融

- **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]

👆One of the early explainability methods that use Gradient Tracing

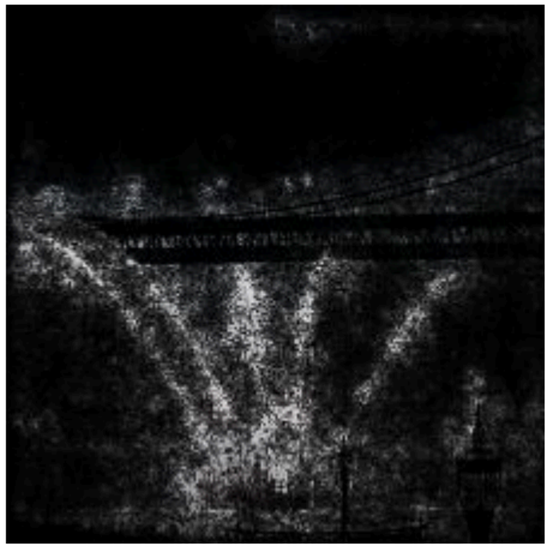
Integrate the gradients along a **straight-line path from baseline to input**

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

Original image

the input

Integrated Gradients



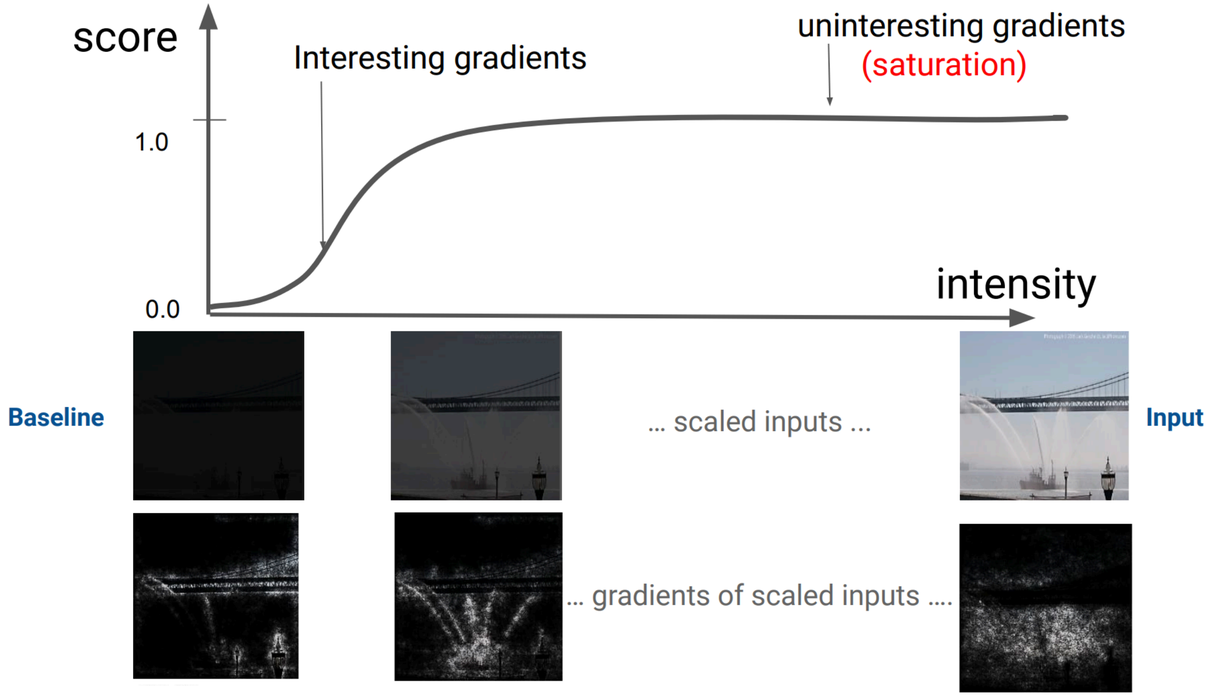
compare to the naitve, the image is better

important thing is that it has some form of base.
You have a baseline to compare

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**

no need to do from scratch,
library available



Why is this Image labeled as a “clog”?

Original image

“Clog”

Why is this Image labeled as a “clog”?

colour dsitribution between these 2 are similar

Original image

Integrated Gradients
(for label “clog”)

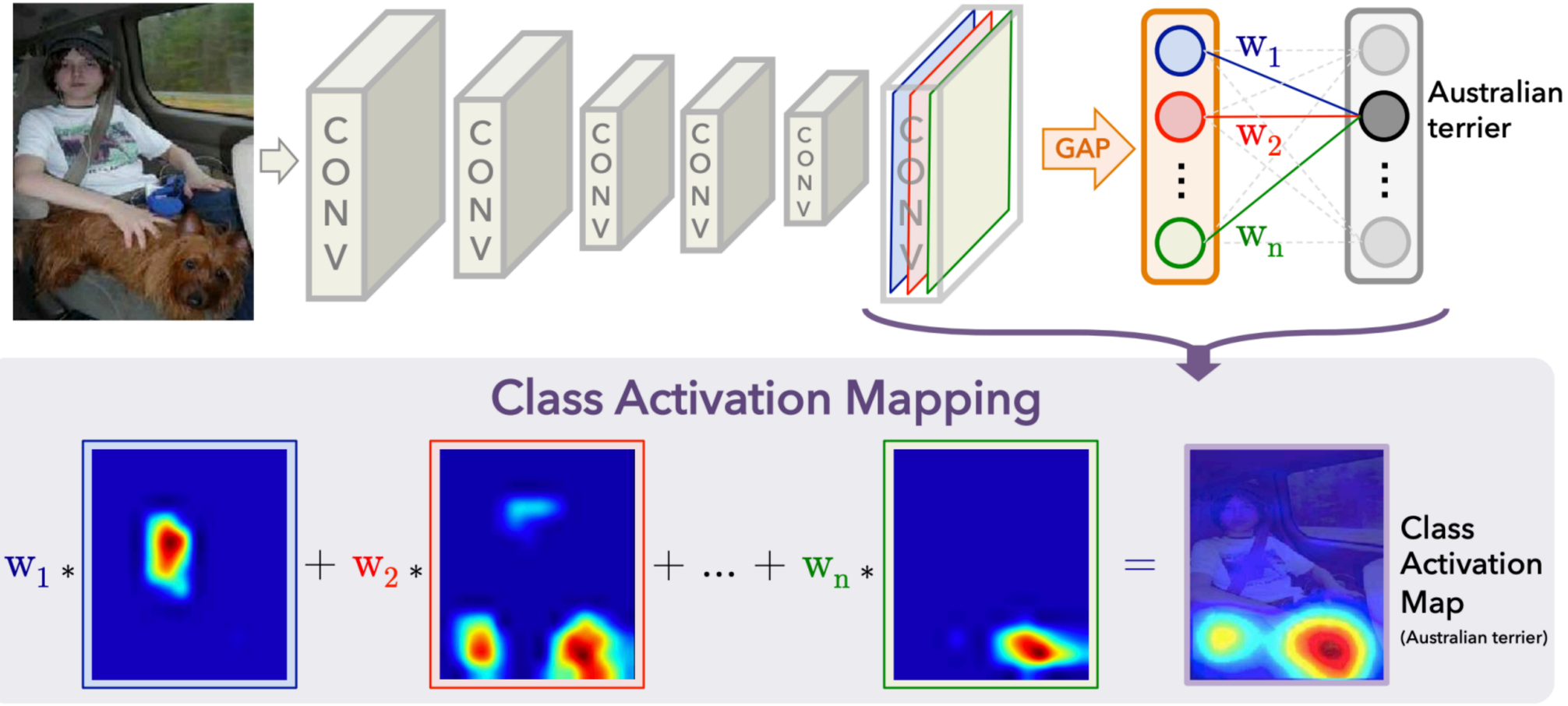
“Clog”

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

Class Activation Mapping (CAM)

One of the methods built on Integrated Gradient

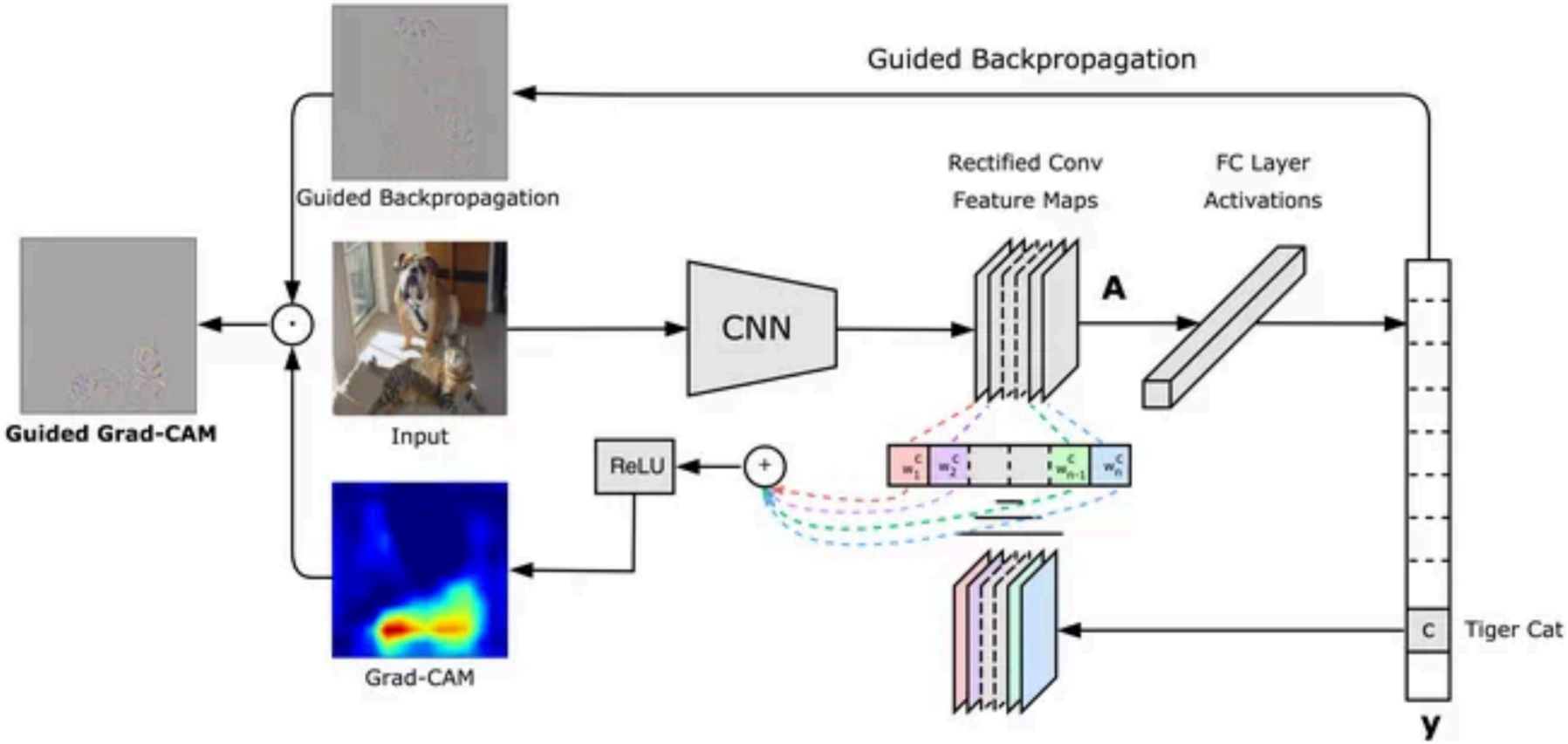
basically skipped



Grad-CAM

take convolutional filers, use as weights for the gradients.
We do not just want final labels but also last filers

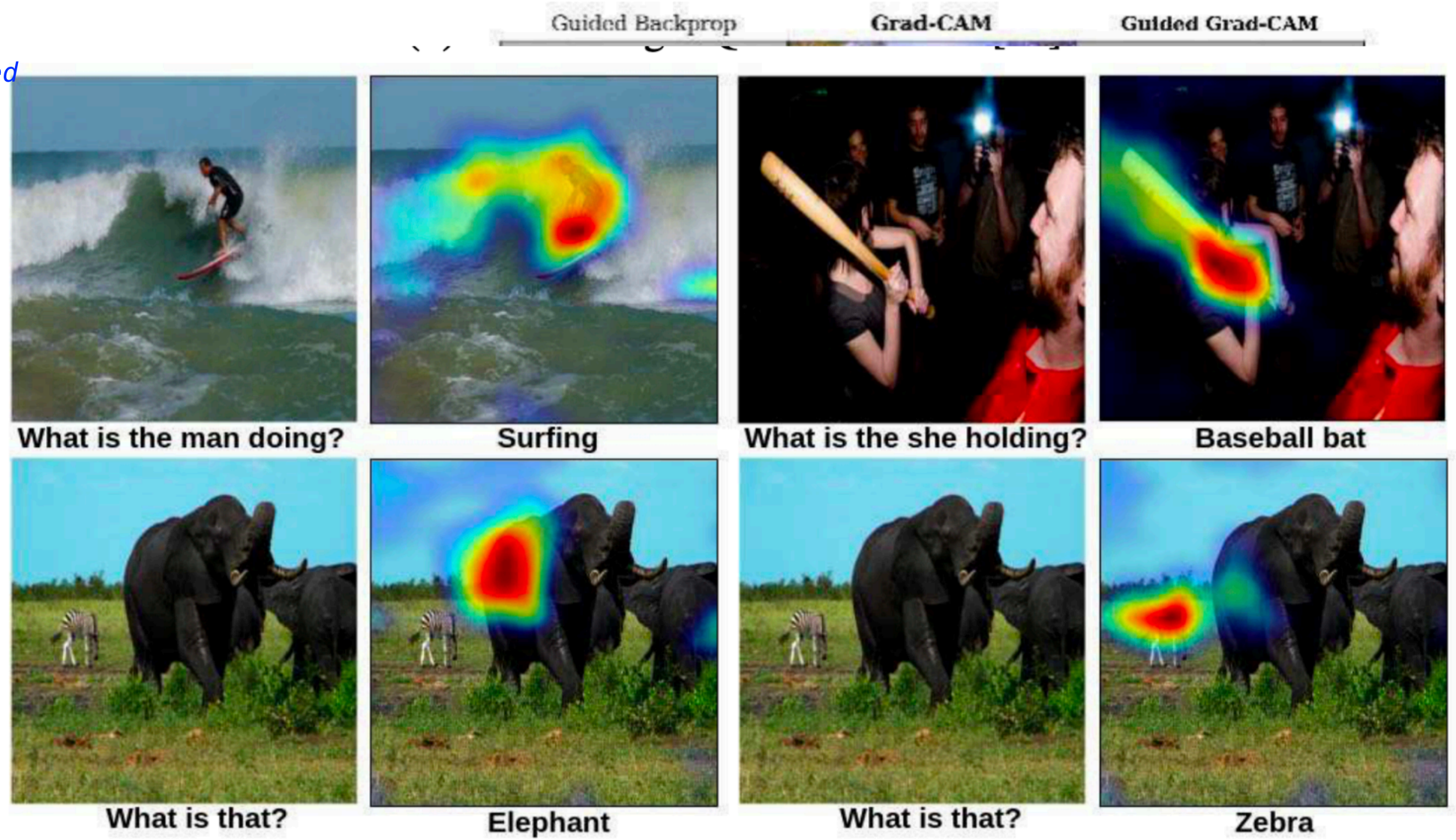
one of the CAM



Grad-CAM VQA

local heat map

Basically Skipped



Gradient Tracing Methods

the evolution of this method

Gradient Tracing Summary

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

httP://captum.ai/
library `captum`

@GPT: 用于分析和理解复杂机器学习模型中的梯度流动和计算过程。
它通过跟踪梯度的传播路径，帮助研究人员和工程师诊断模型中的问题，如梯度消失或梯度爆炸等。
在NLP中，梯度追踪尤为重要，
因为NLP模型（如RNN、LSTM、Transformer等）通常具有复杂的结构和参数依赖。

@GPT: Probing is the technique of analyzing neural network representations by training simple classifiers to determine if they encode specific linguistic or task-related properties.

假设我们有一个训练好的语言模型（如BERT），我们想知道它的隐层表示是否包含词性信息。
为此，我们可以对每个词提取模型某一层的隐层表示，
然后训练一个简单的分类器（探测器）来预测每个词的词性（如名词、动词、形容词等）。
如果分类器能准确预测词性，这表明该层的表示确实包含了词性信息。
这就是一个具体的探测（probing）任务的例子。

Probing

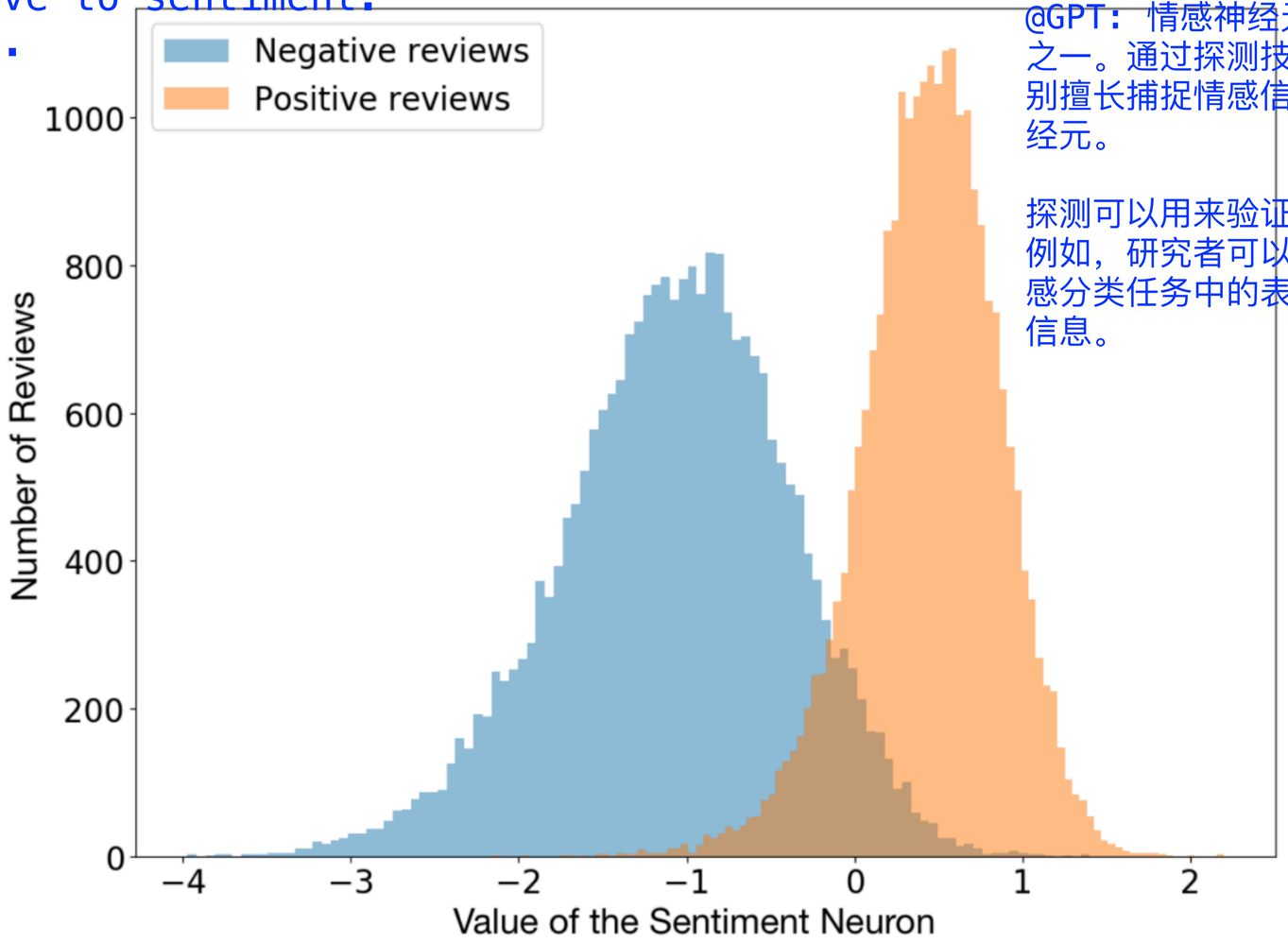
a more modern approach

The Sentiment Neuron

motivation of probing

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.

one feature out of all (e.g. 798) features encodes 95% of decisions.
it is extremely sensitive to sentiment.
called sentiment neuron.



@GPT: 情感神经元可以看作是探测分析的具体发现之一。通过探测技术，研究者可能发现某些神经元特别擅长捕捉情感信息，从而标记这些神经元为情感神经元。

探测可以用来验证和解释情感神经元的存在和功能。例如，研究者可以用探测分类器测试这些神经元在情感分类任务中的表现，以确定它们确实捕捉到了情感信息。

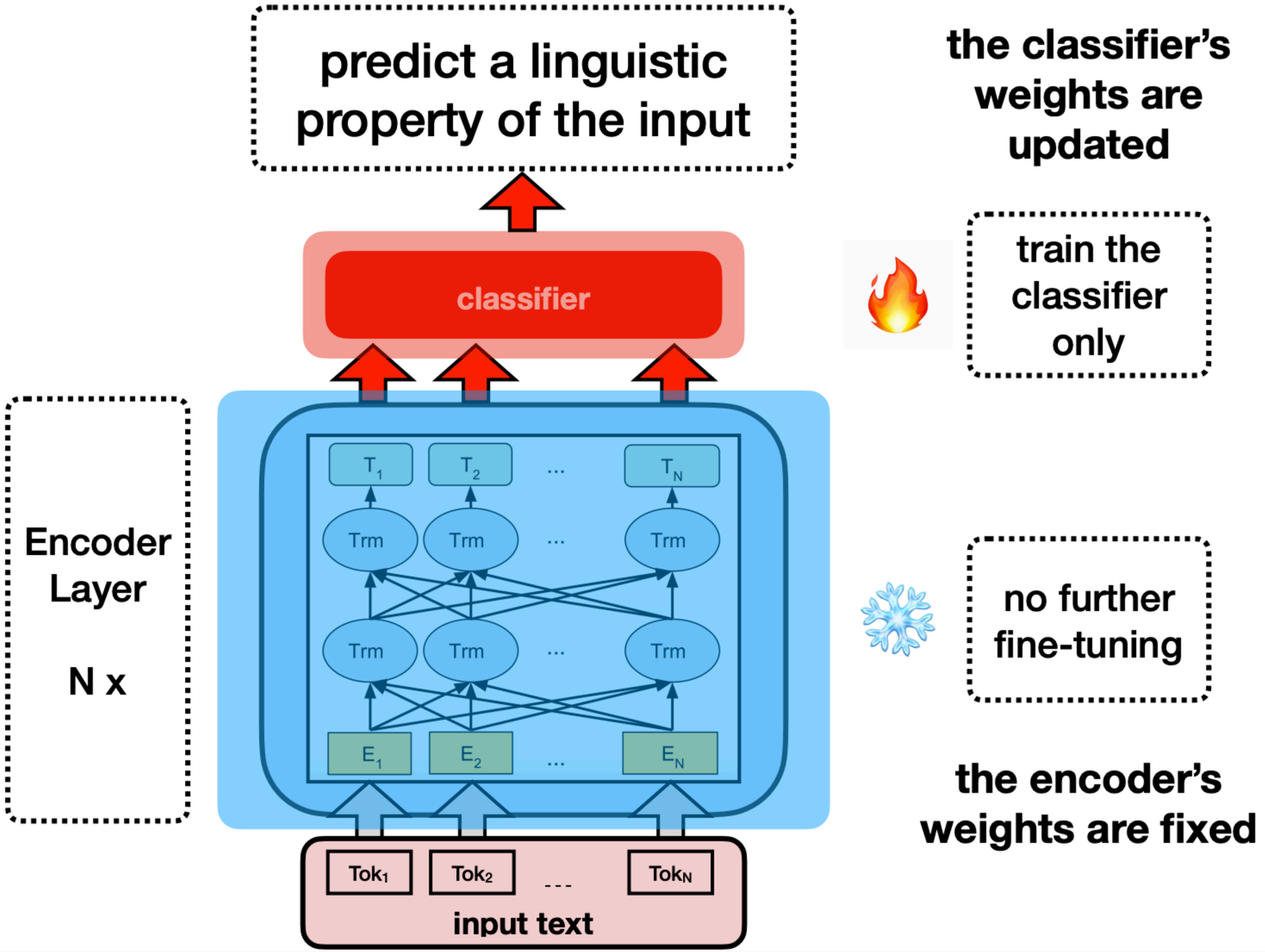
Linguistic Probing Tasks

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

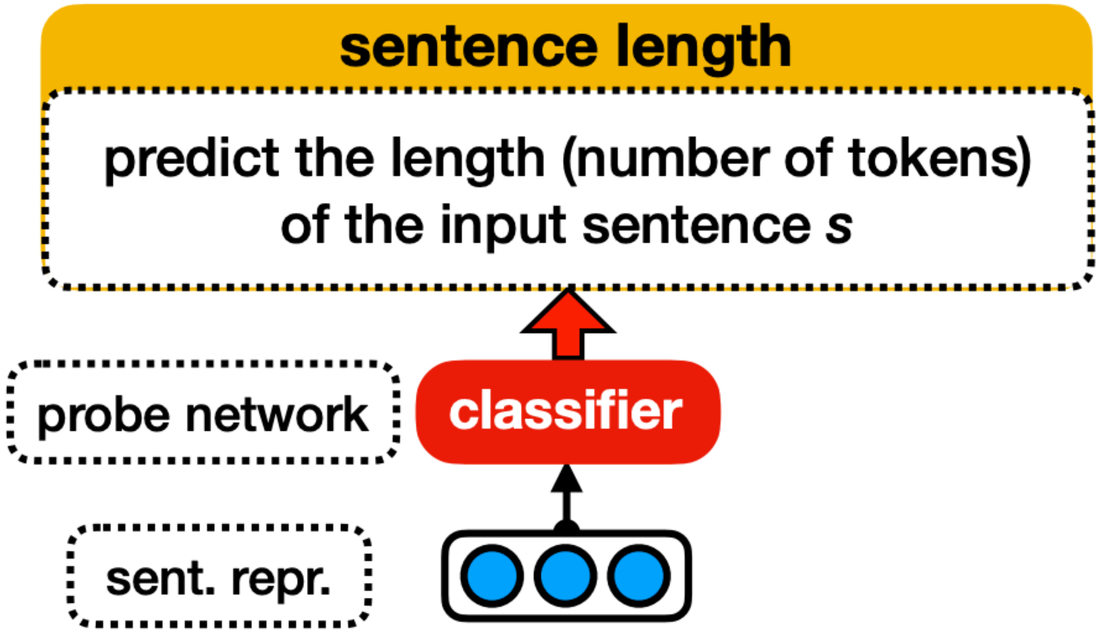
Probing Tasks

- 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 不可知论 with respect to the encoder architecture, as long as it produces a vector representation of input text
- does not necessarily correlate with downstream performance

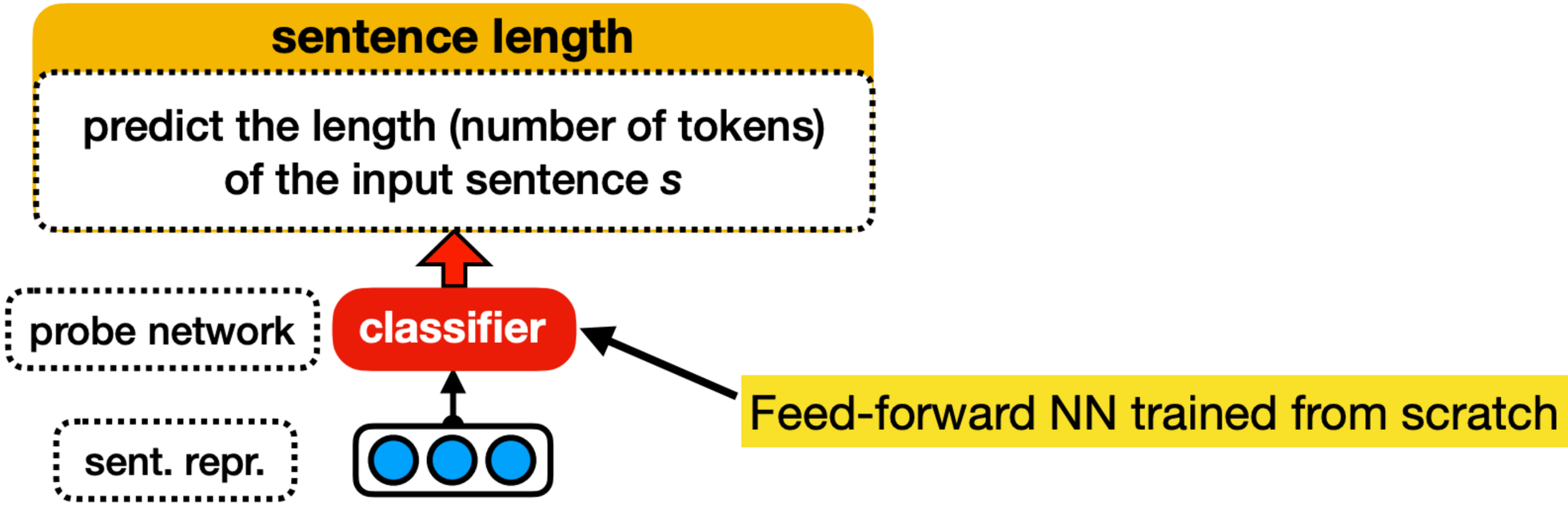
Probing Architecture



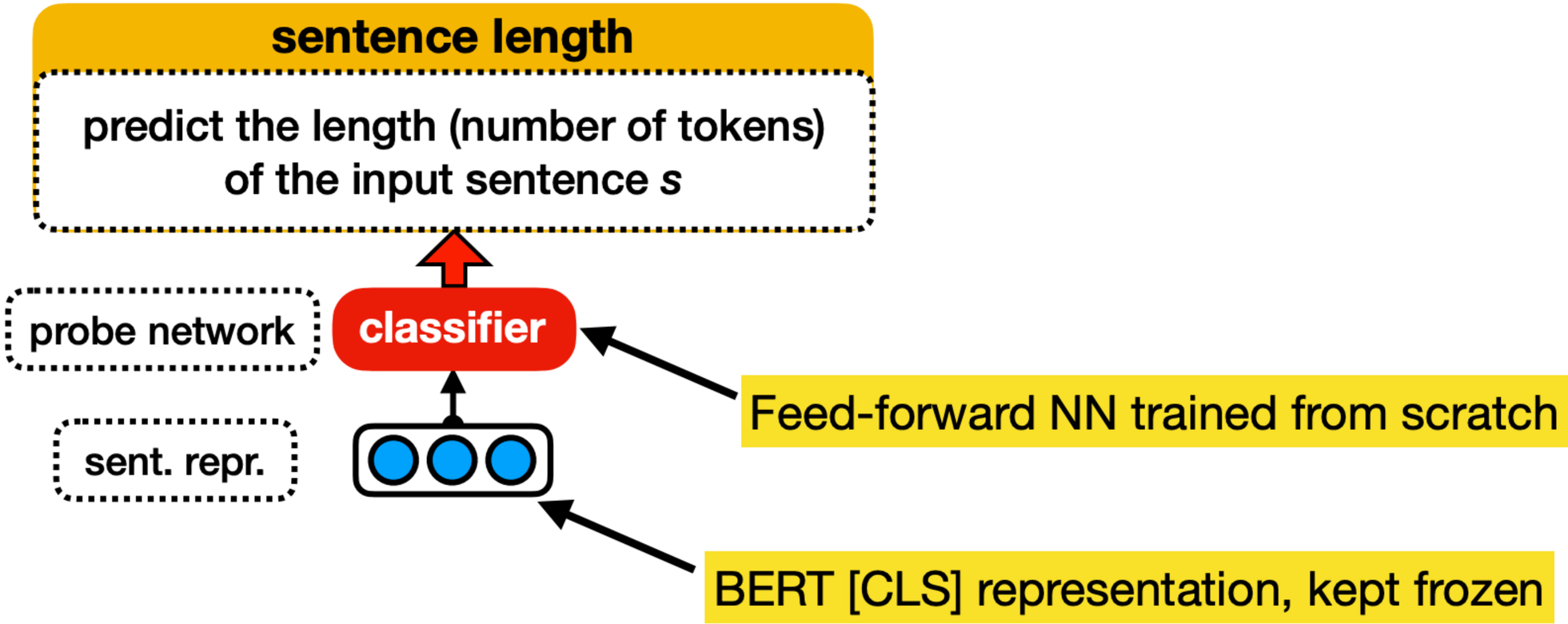
Types of Linguistic Probing Tasks



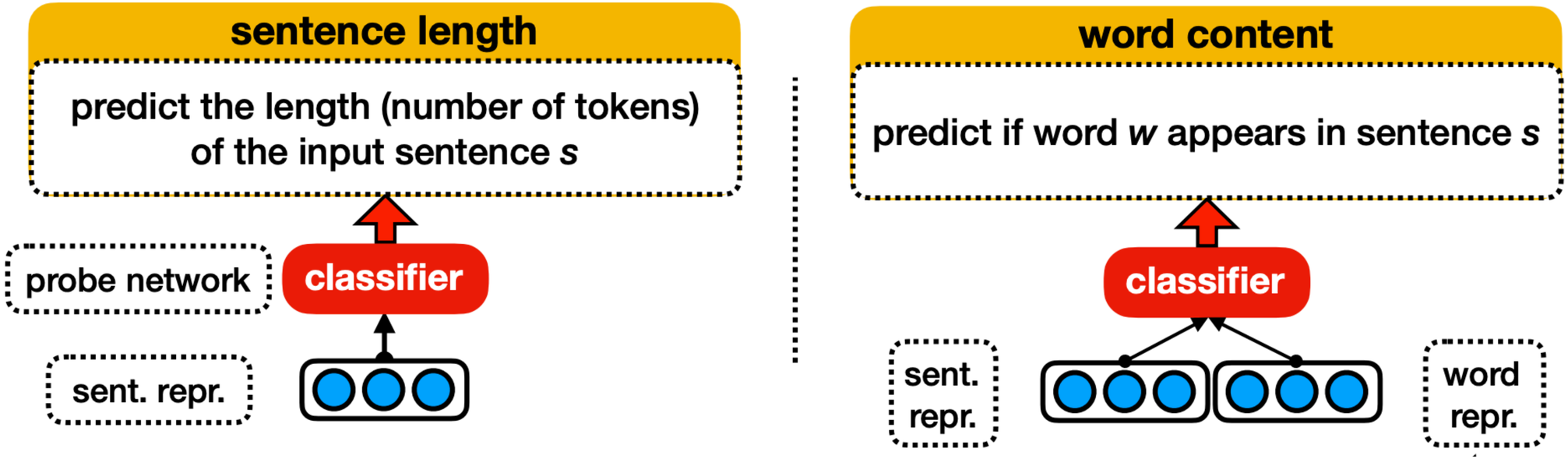
Types of Linguistic Probing Tasks



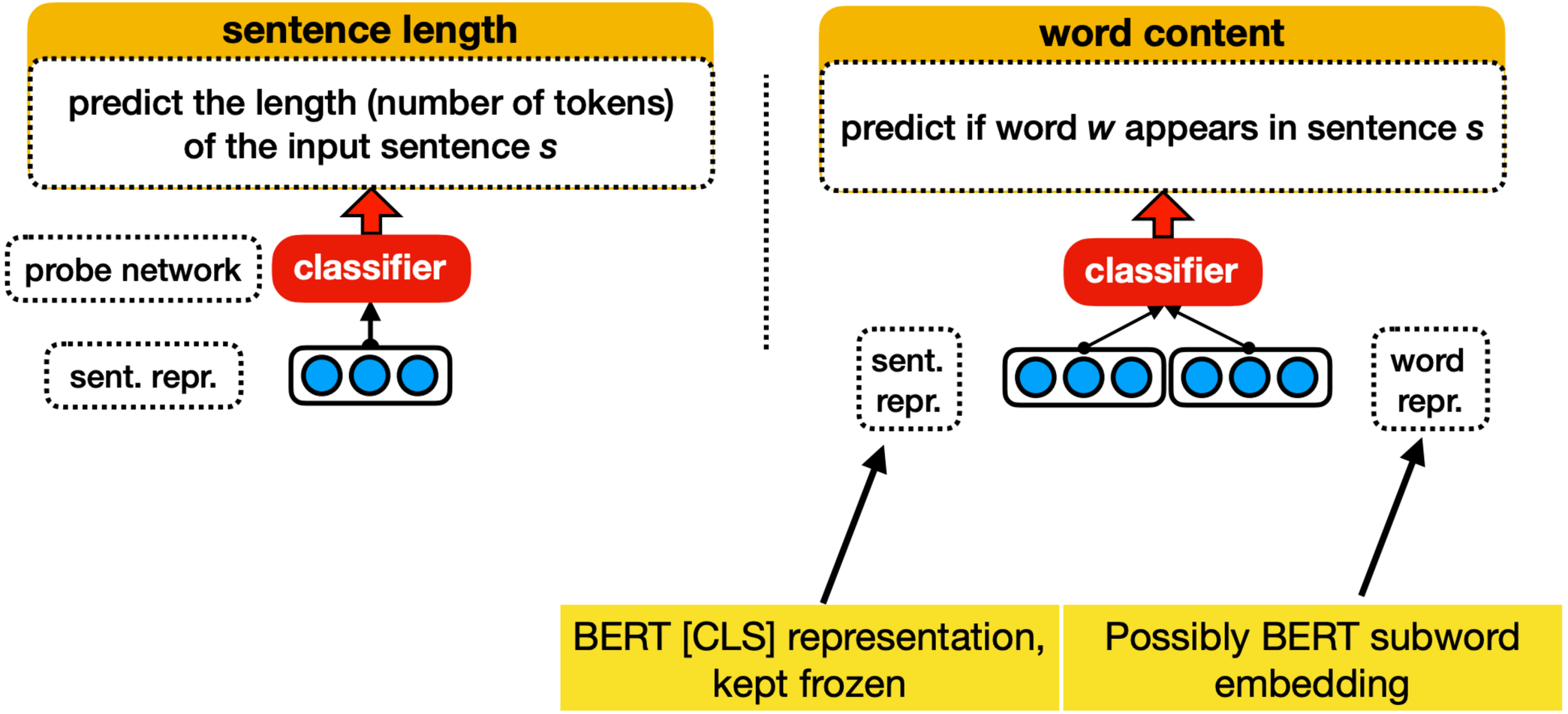
Types of Linguistic Probing Tasks



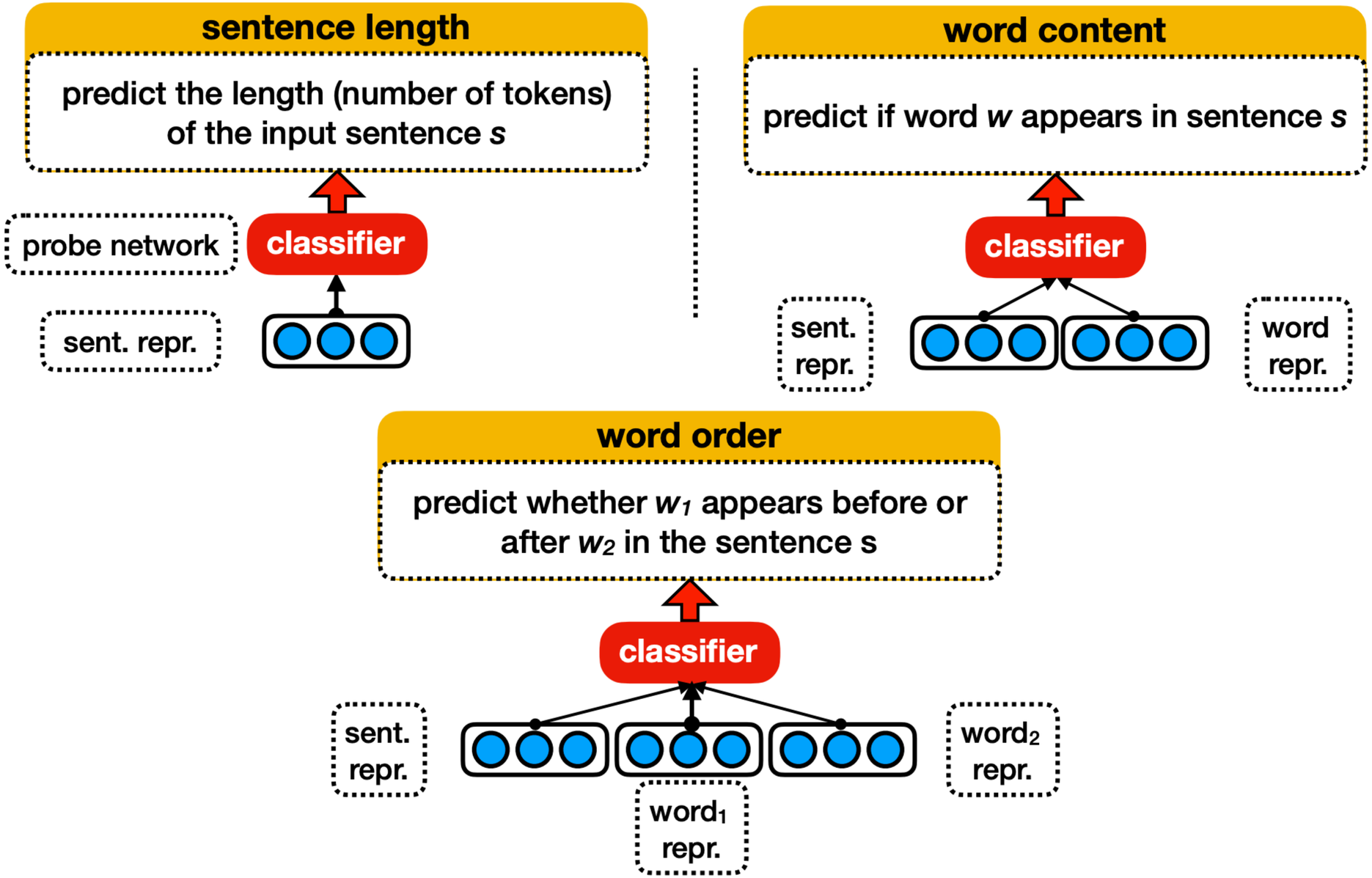
Types of Linguistic Probing Tasks



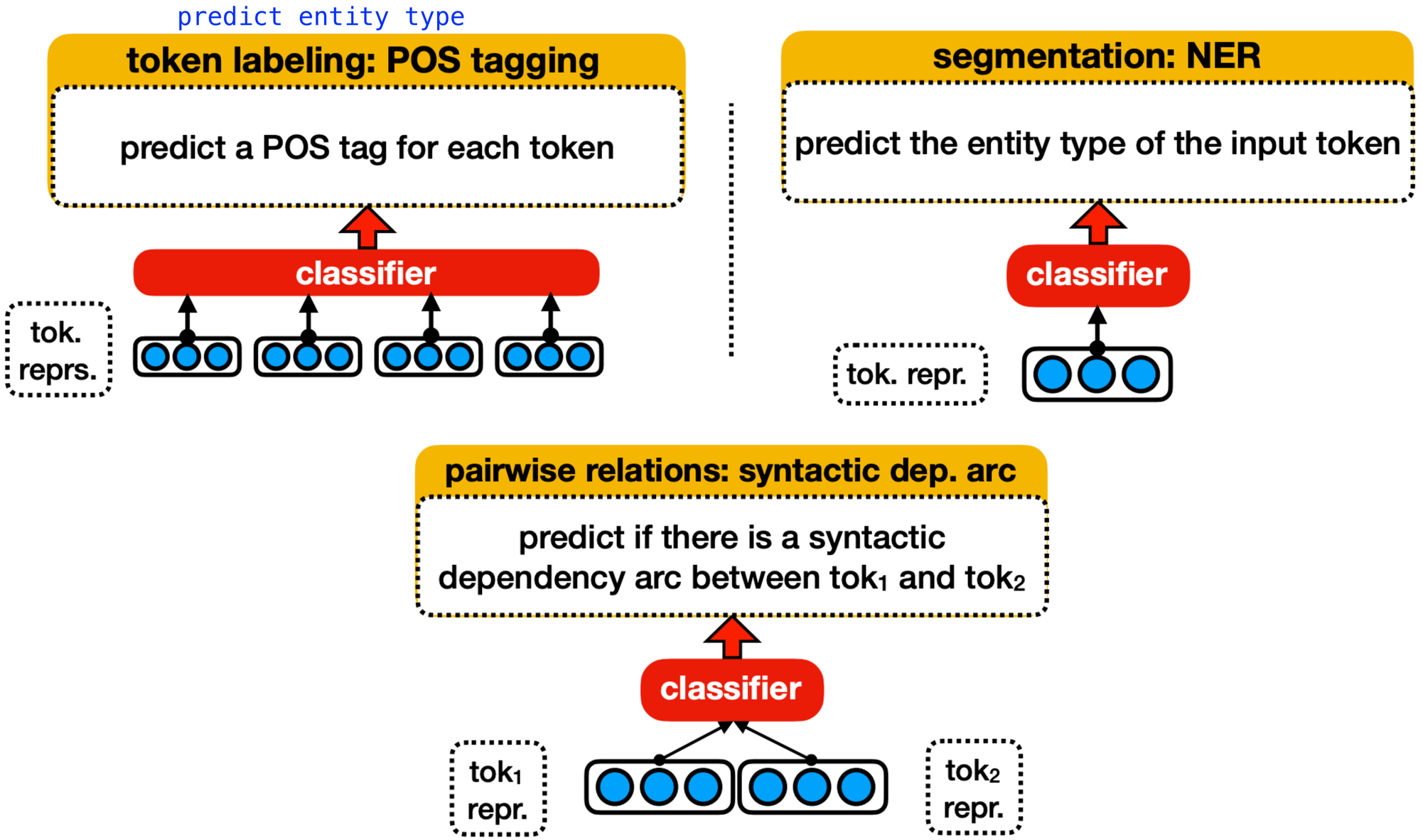
Types of Linguistic Probing Tasks



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

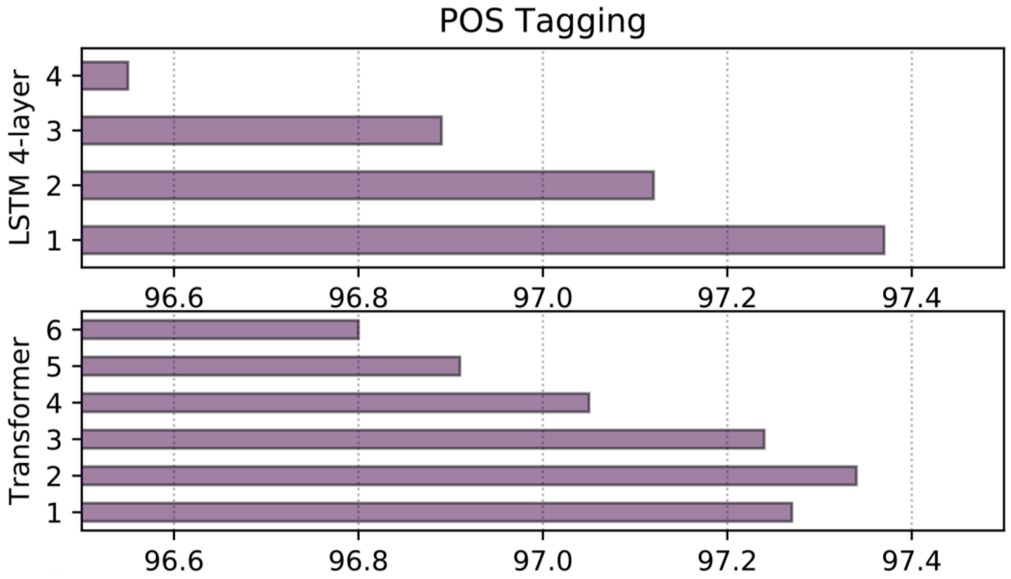
- if we cannot train a classifier to predict a property of the input text based on its representation, it means

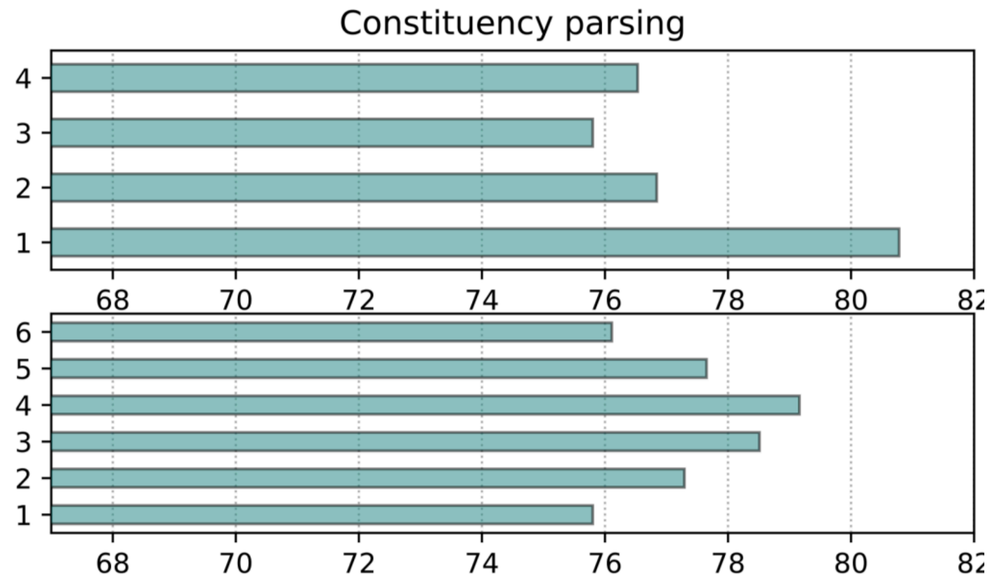
Nothing?!

or classifier messed up

if the classifier works well, then i have good features.

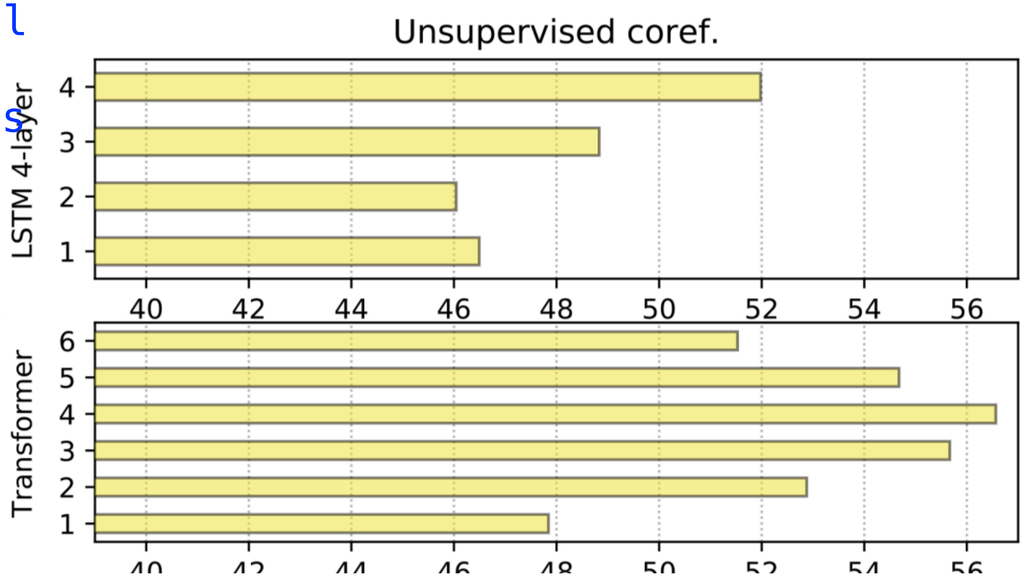
Early Layers Capture Local Syntax, Later Layers Focus on Content





look at the layers of the model

look at a concrete task such as pos tagging for my probe.
It seems like POS tagging is solved pretty well by early layers of the model and not so much by laters



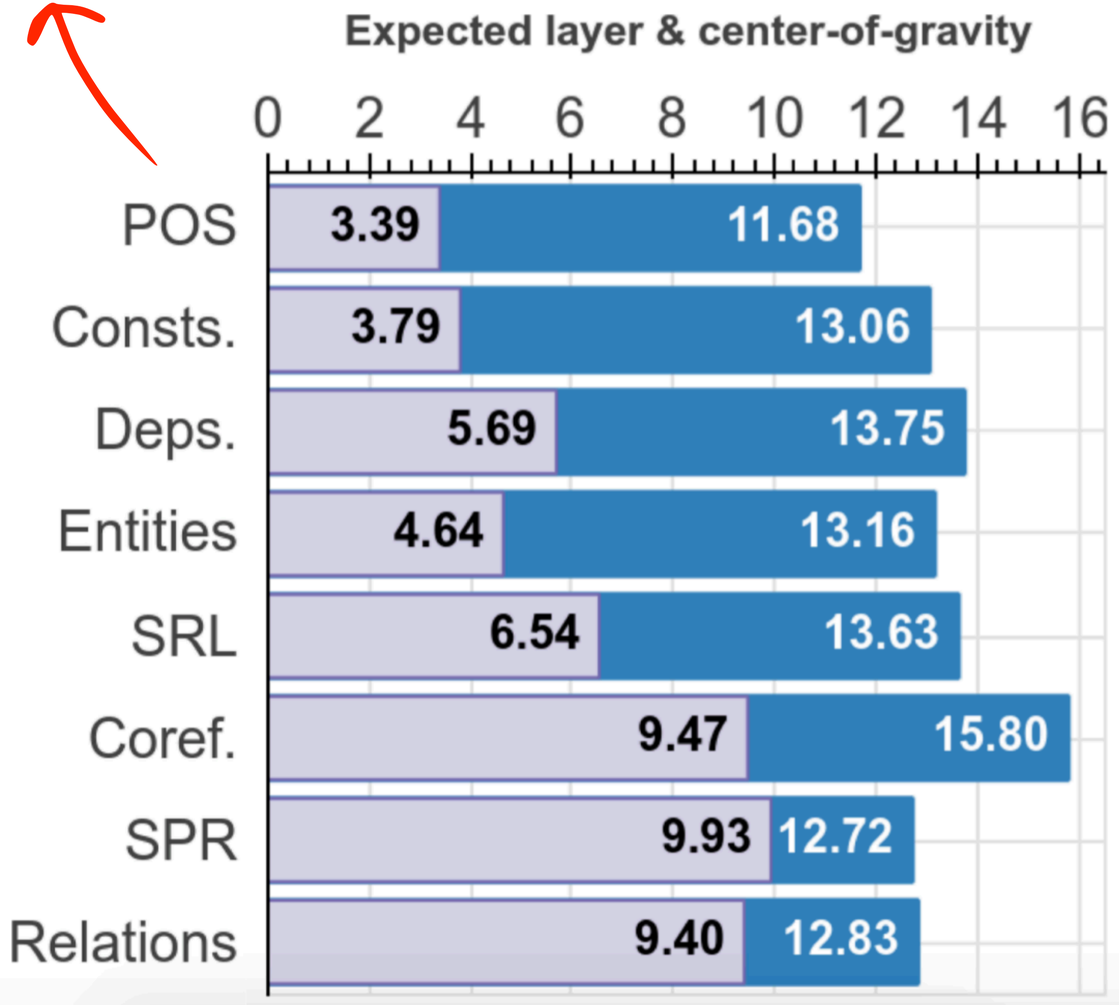
(Peters et al., 2018)

Localizing Task-Specific Computations

Again. POS seems to happen earlier, relations later etc.

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



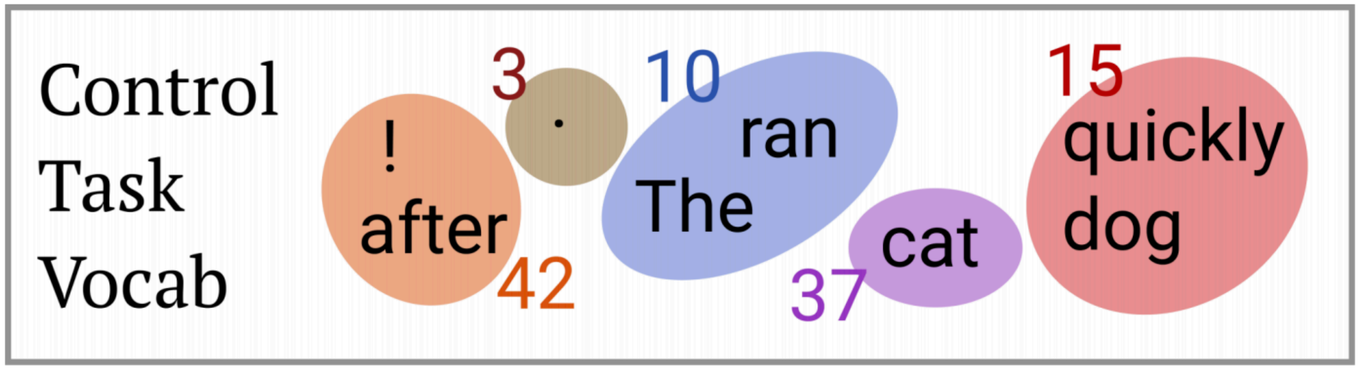
Control Tasks [Hewitt et al., 2019]

an interesting twist,
just random numbers that have no linguistic properties

- 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)

how good are you at finding these made-up words, how good are you at predicting these randomness.

set up `The` is always 10.



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**

在探测任务（probing tasks）中，控制任务（control tasks）是一种对照实验，
用于验证探测结果的可靠性和模型表示的真正能力。
具体来说，控制任务帮助确定模型的表现是否真正反映了所探测的特性，而不仅仅是利用数据中的某些统计特征或偏差。

1. 主探测任务：

假设我们进行一个词性标注探测任务，训练一个探测分类器来预测每个词的词性（名词、动词、形容词等），并且模型表现良好。

2. 控制任务：

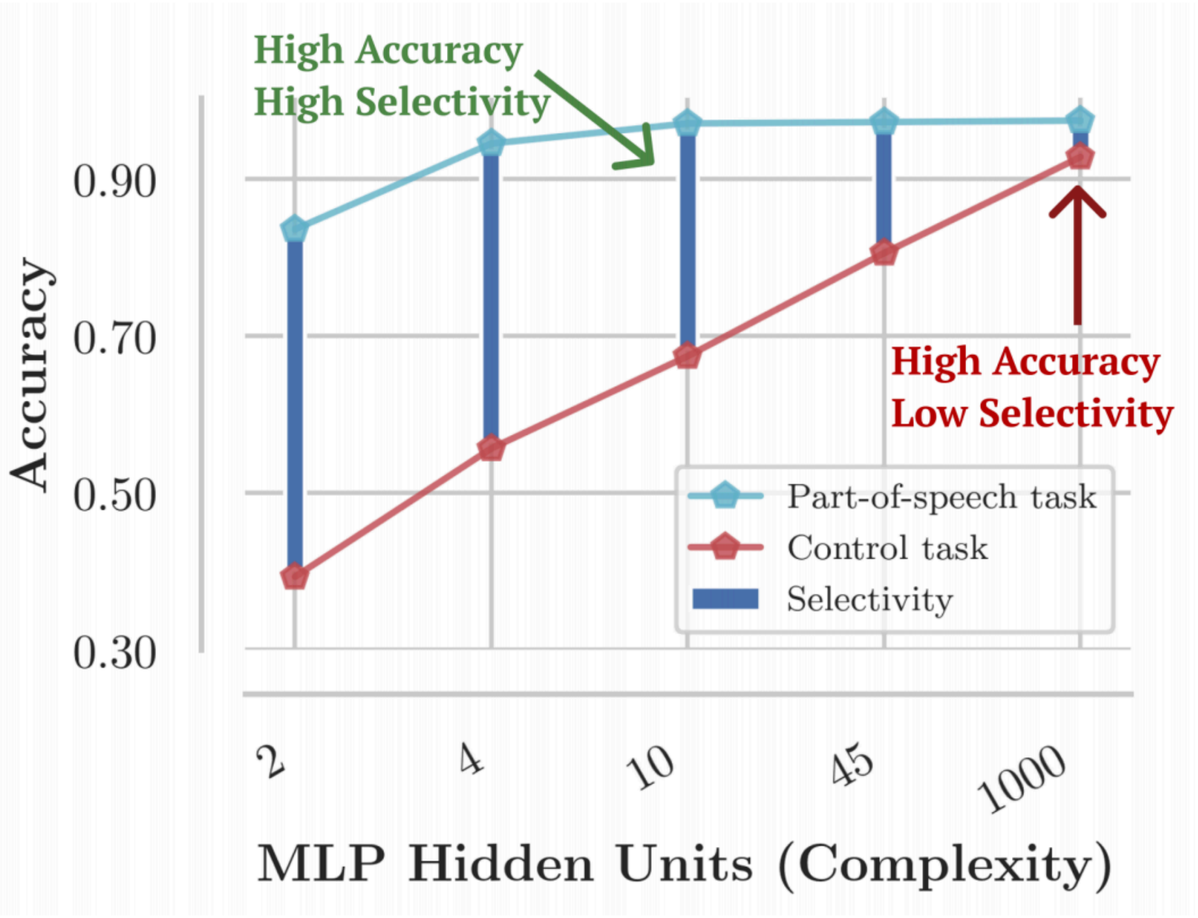
为了确保模型的高性能不是由于数据中的简单模式或偏差，我们设计一个控制任务，例如随机打乱词性标签。
我们使用相同的模型表示和相同的探测分类器架构，但这次分类器的目标是【预测这些随机打乱的标签】。

3. 评估：

如果探测分类器在控制任务中的表现明显低于主探测任务，这表明模型的高性能确实源于捕捉了词性信息，而不是因为数据中的某些统计模式或偏差。
如果控制任务的表现与主探测任务相似，那么我们需要重新评估探测结果，因为这可能表明模型的表示并没有真正捕捉到词性信息，而是利用了数据中的某些简单特征。

Probe Selectivity

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



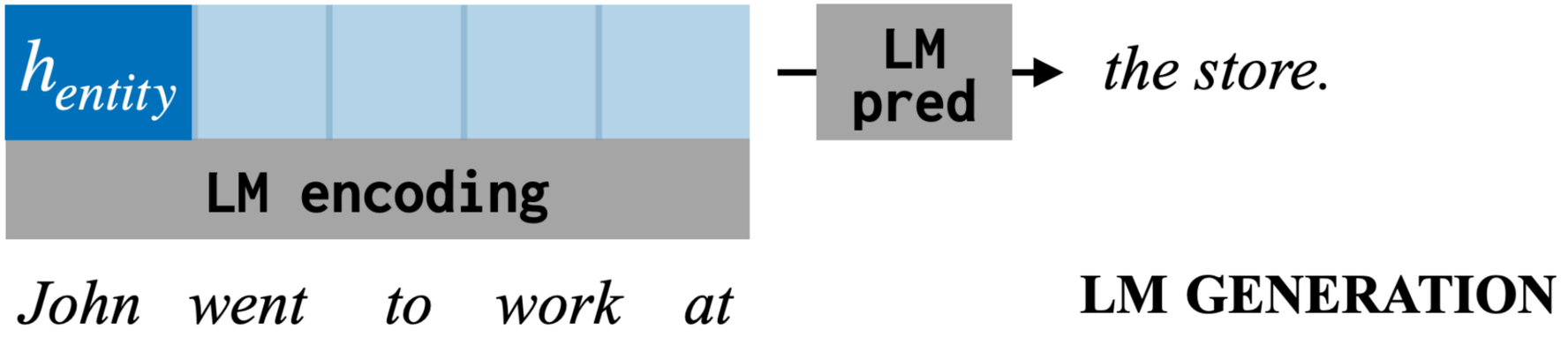
Probing for a Better Tomorrow

- 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

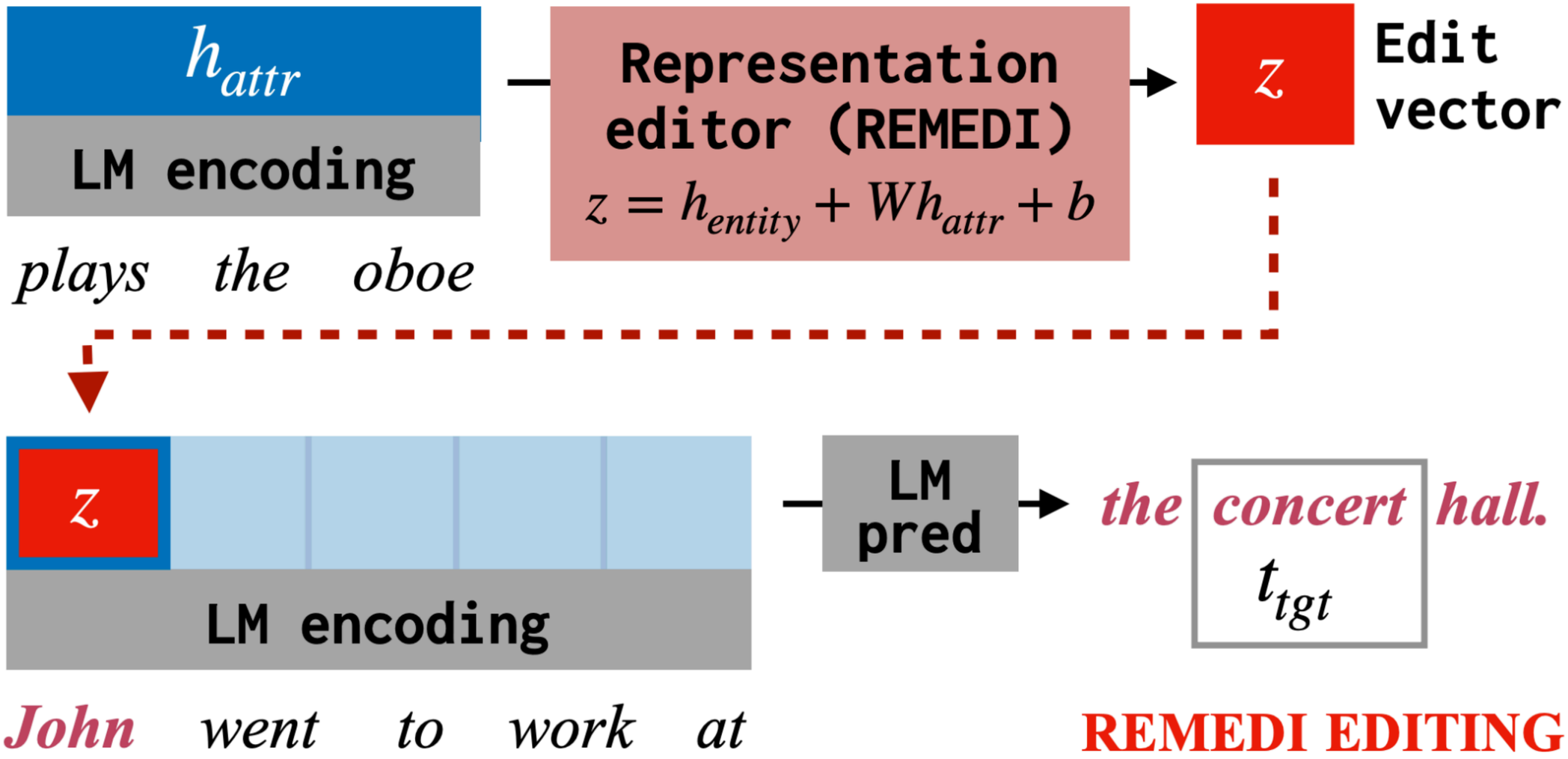
explicitly editing around the model



prompt: john went to work at,
and the model predicts: `the store`

Knowledge Editing

adding extra content to `john` and
loading them into the `john` token



Knowledge Editing

so now get very different behaviours and can lead the generation to a certain way

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