CSE 156 | Lecture 15: Interpretability of Neural NLP

Ndapa Nakashole

November 19, 2024

Administrative matters

- **PA3**: was due yesterday, today last day to submit with a 5% penalty
- **PA4**: Released yesterday, Nov 18 due on Dec 2
- **Quiz 2**: due on Thursday, Nov 21
- **Quiz 3**: out on Thursday, Nov 21

Today

1. Interpretability Overview
2. Probing Classifiers
3. Sparse Autoencoders
4. Dataset Artifacts
5. Other Considerations: Reasoning when the context is long

The Problem

- LLM and other ML models have become highly capable
- **Problem**: they make decisions for reasons we do not understand.

- **Why is this a problem?**
  - Concern over **untrustworthy** models in widespread use in the economy and in our lives (Ngo et al., 2022)

Interpretable: Better understanding $\rightarrow$ more trustworthy

Interpretability

Hypothesis: Models learn interpretable algorithms that can be understood by humans, but we must learn to make them legible

Can uncover:

- **Features**: The variables that the model uses
- **Algorithms**: The algorithms learned to map from features to outputs

Interpretability

Understanding what happens inside ML models

Interpretability: A microscope to see what LLMs are "thinking"

Exploring Gemma Scope

An Introduction to AI Interpretability and the Inner Workings of Gemma 2 2B

👋 HELLO!

The inner workings of modern AIs are a mystery. This is because AIs are language models that are **grown, not designed**.

The science of understanding what happens inside AI is called interpretability.

This demo is a beginner-friendly introduction to interpretability that explores an AI model called Gemma 2 2B. It also contains interesting and relevant content even for those already familiar with the topic.

🔧 GET STARTED

!

👩‍🔬

START HERE

Microscope

Scan Gemma 2's brain to see what it's thinking.

Analyze Features

Make features fire and figure out what they do.

Steer Gemma

Change Gemma's behavior by manipulating features.

Historical Perspective: models were small and interpretable

Binary Logistic Regression

$$p(y = 1 \mid x) = \text{sigmoid}(wx)$$

Sentiment Analysis with Binary Logistic Regression

$$p(y = 1 \mid x) = \text{sigmoid}(wx)$$

- Classify text as **positive** $(y = 1)$ or **negative** $(y = 0)$.
- **Features** (x): Presence of specific keywords:
  - x_1: "good"
  - x_2: "bad"
  - x_3: "excellent"
  - x_4: "poor"
  - x_5: "okay"
- **Weights** (w):

$$w = [2.5, -3.0, 1.5, -1.0, 0.0]$$

Multi-class Logistic Regression (Softmax)

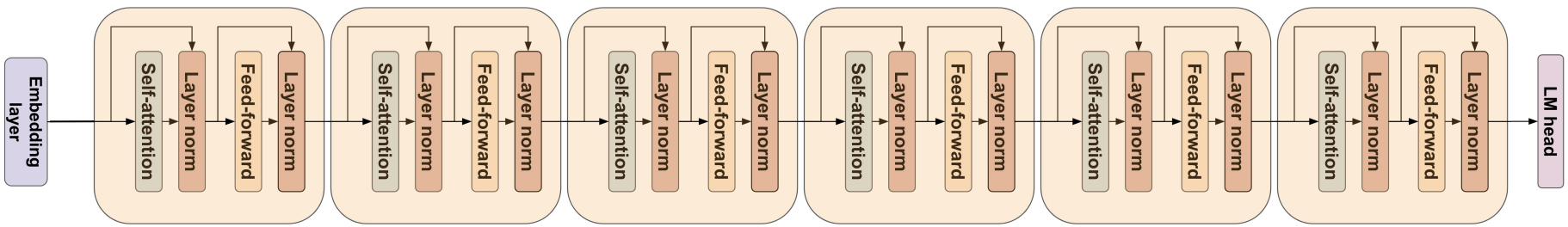
Is also interpretable!

Softmax Logistic Regression

$$p(y = k \mid x) = \text{softmax}(\boldsymbol{W}x + b)_k$$

- Each class has its own set of weights (rows of $\boldsymbol{W}$).

Now models are big



Consider BERT: Token Embeddings

["The", "movie", "was", "not", "bad"].

- Embedded representation:

$$x_{\text{embed}} = [E[t_1], E[t_2], \ldots, E[t_n]]$$

- Dense, high-dimensional embeddings (e.g., 768 for BERT-base).

Challenges:

- Token embeddings are not human-readable
- Representations entangle semantic and syntactic information

Consider BERT: Multi-Head Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Challenges:

- 12 attention **heads** aggregate information differently
- Information is distributed across **layers** (e.g., 12 layers in BERT-base)
- No direct attribution to input features

BERT: Output Layer

Final Output:

$$y = \text{softmax}(W_{\text{output}} h_{\text{last}} + b_{\text{output}})$$

- h_{last}: Contextualized representation from the last layer

Challenges:

- Contextualized embeddings entangle information across tokens and layers
- No clear, interpretable mapping between h_{last} and the input tokens

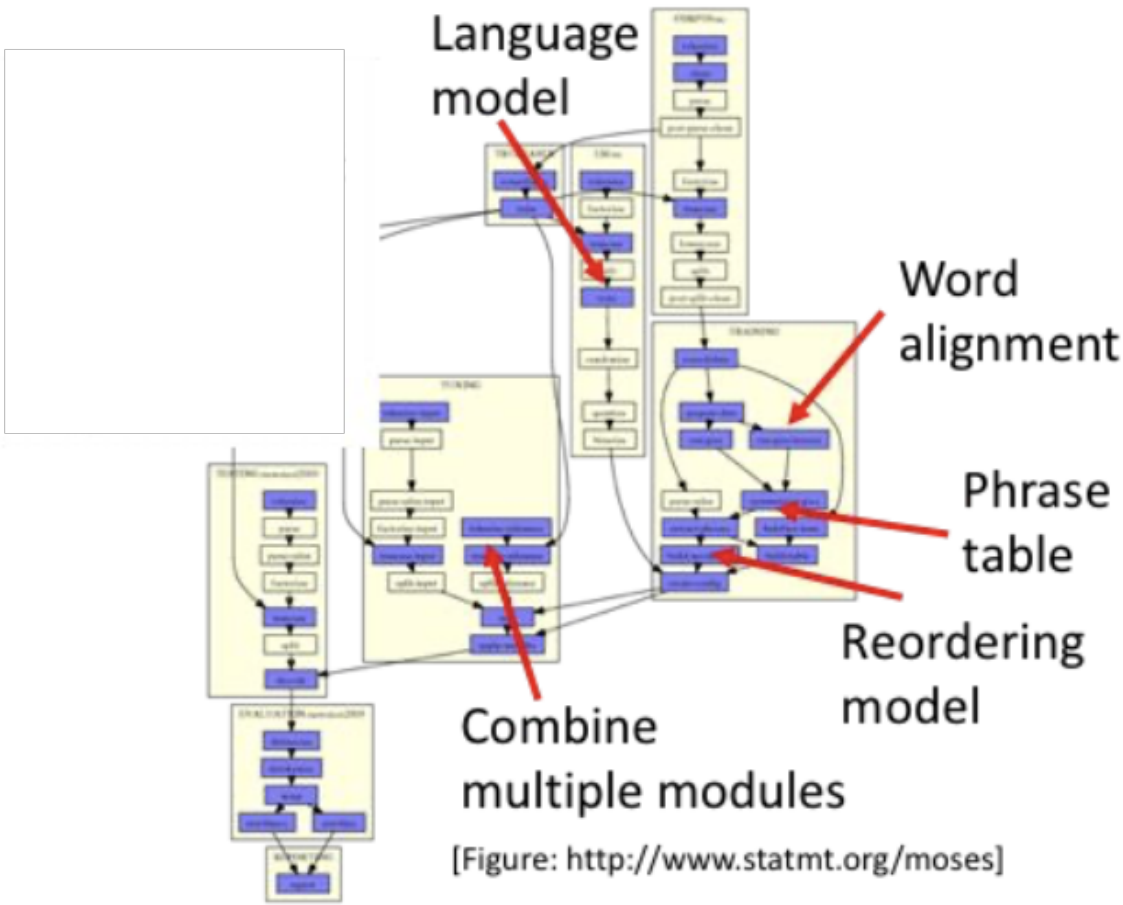
Sentiment Analysis: Logistic Regression vs. BERT

- Logistic Regression:
  - Direct mapping from features to weights
  - Positive/negative weights are interpretable
- BERT:
  - 110M parameters: High-dimensional embeddings and weights
  - Multi-layered, non-linear transformations
  - Entangled representations across tokens, heads, and layers
- BERT's complexity provides strong performance on NLP tasks
- But sacrifices interpretability

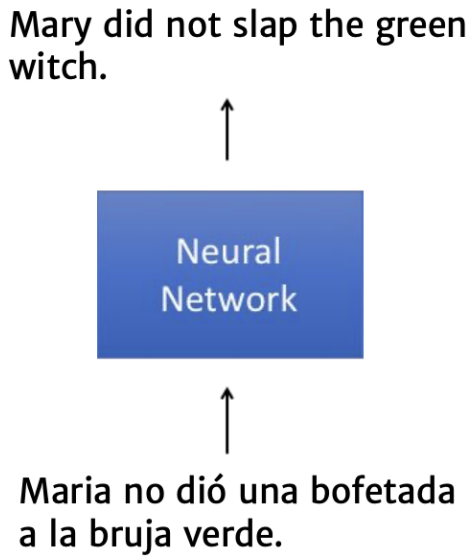
Historical perspective, if models were big, they were modular

- Before seq2seq: **Machine Translation** systems were:

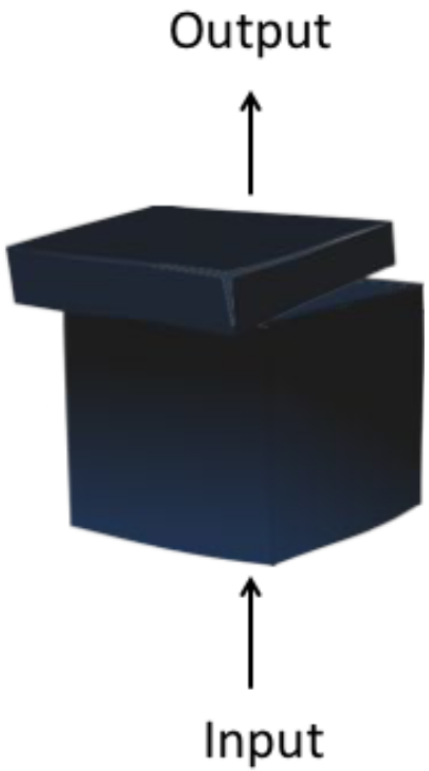
  Complex but **Modular architectures** with human understandable features



[Figure: http://www.statmt.org/moses]



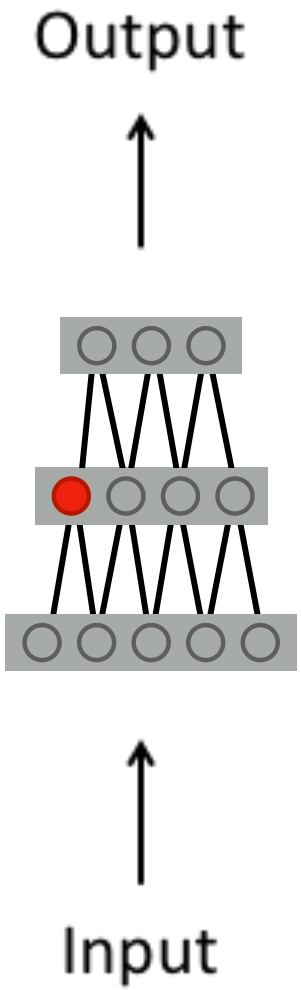
Opening the black box



- What interpretable structures do neural models **learn**?
- Why do they make particular decisions?
- When do they succeed and fail?

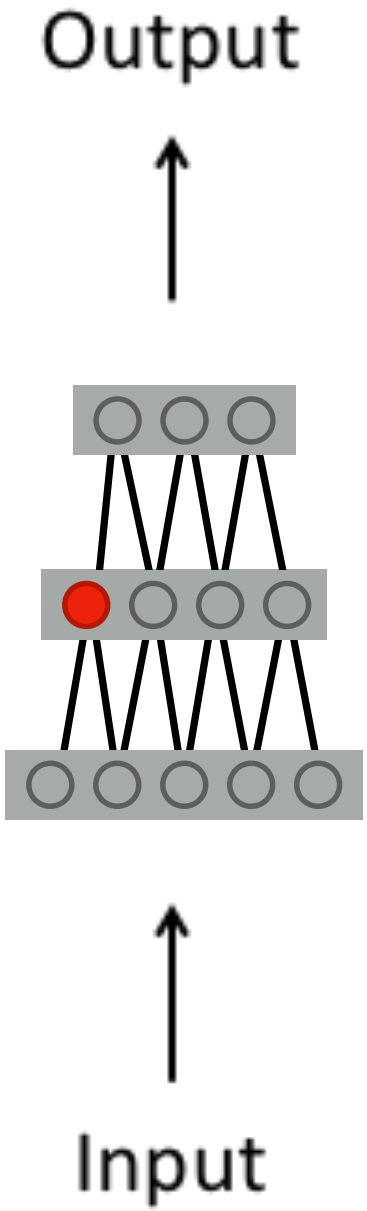
Probing Classifiers

Does the model know about property y?



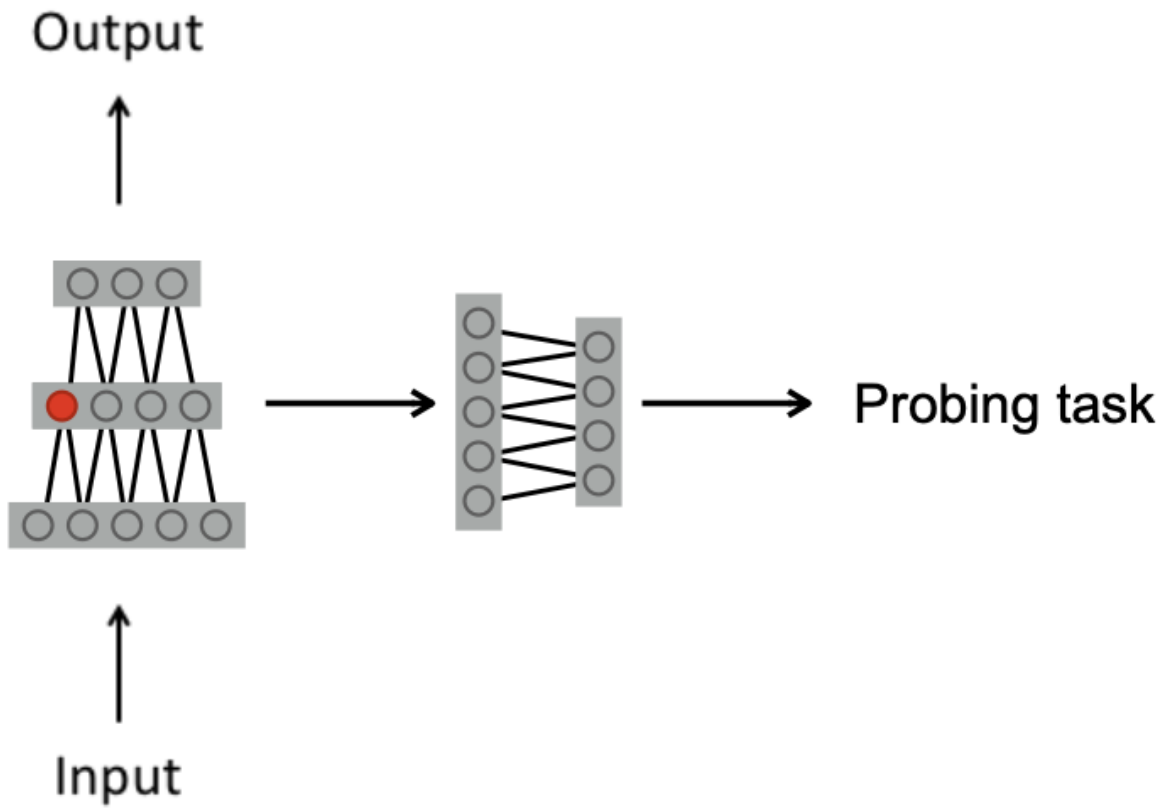
- What is the role of different components (neurons, layers, attention heads)?
- E.g., Does neuron x know something about property y?

Probing



1. Train the model then freeze its parameter
2. Use the model's features as inputs to a simple probe (typically a linear classifier) that predicts y.

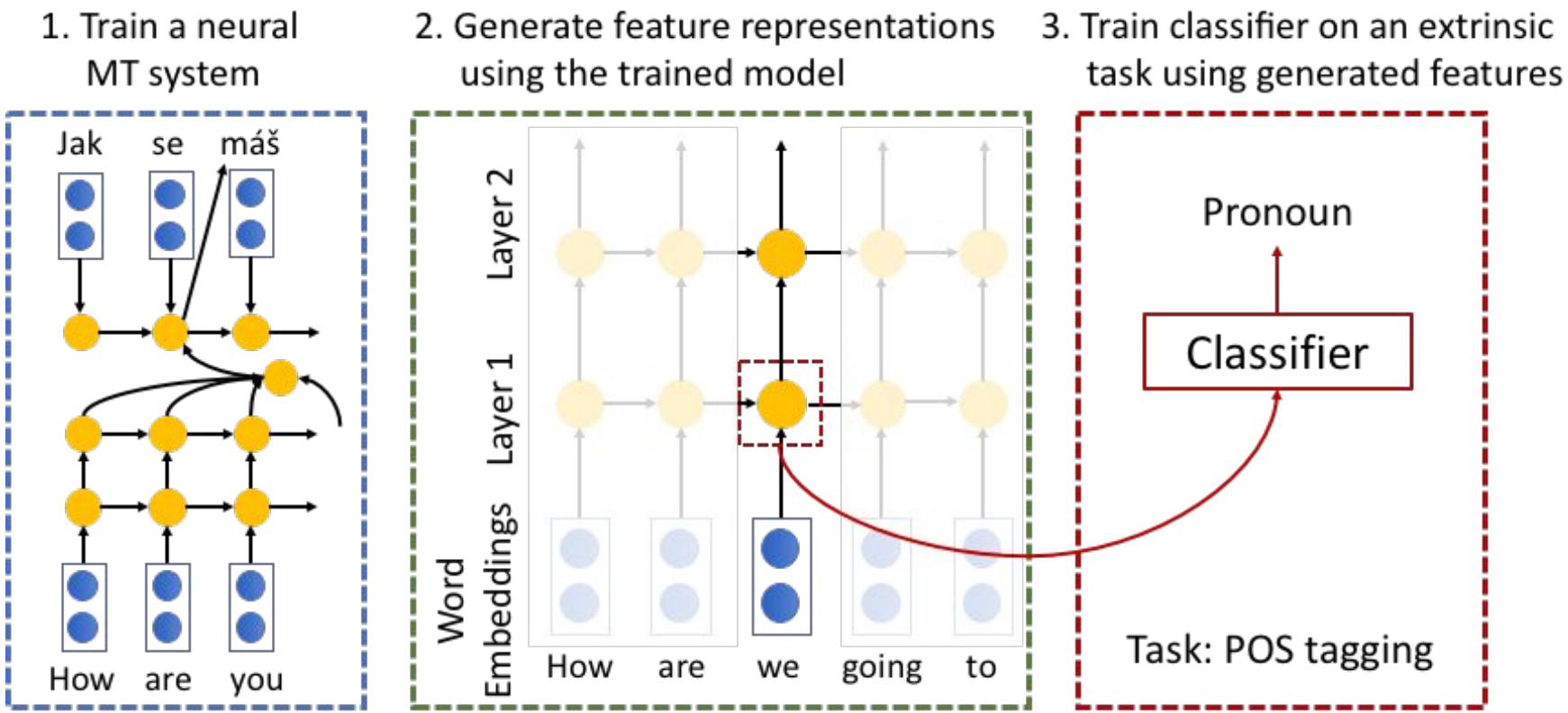
Probing



Probe

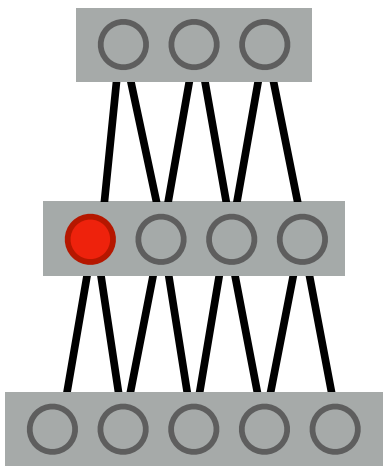
A probe is a classifier that is specifically trained to predict some property from a pretrained model's representations

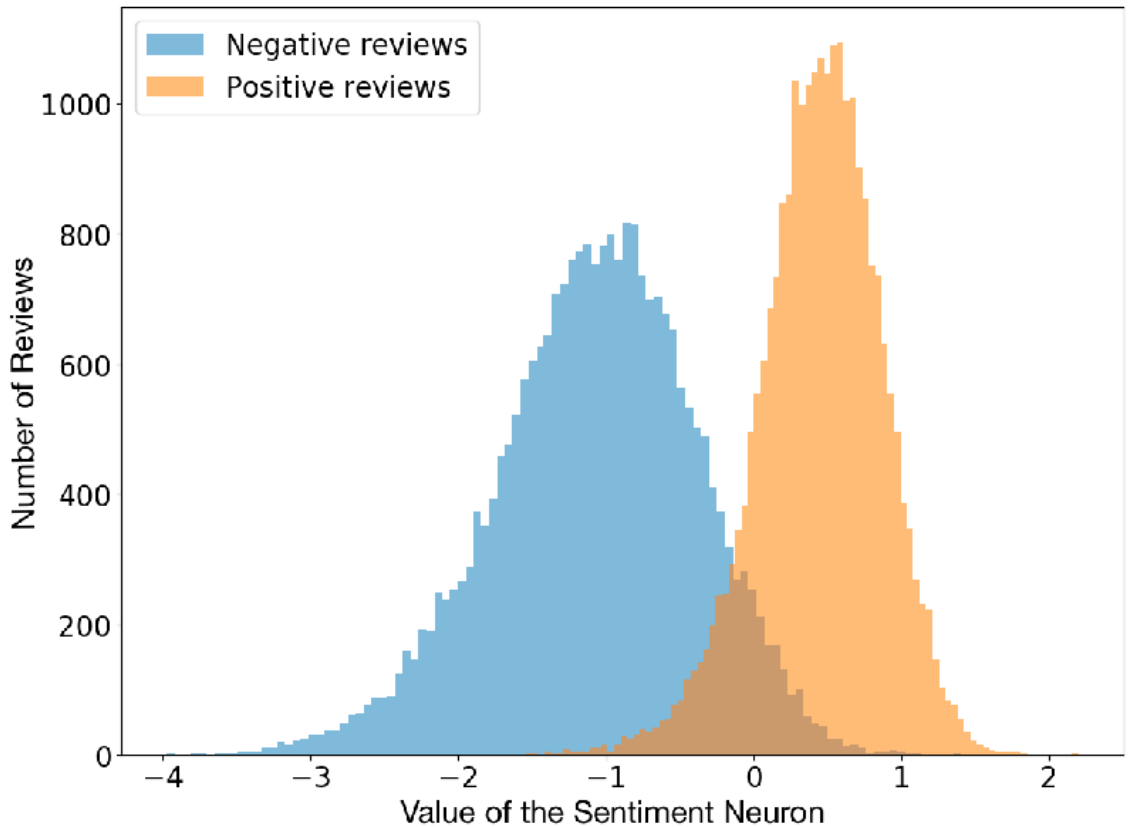
Example: machine translation



E.g., The Sentiment Neuron

- Sometimes the feature of interest is encoded by a single neuron

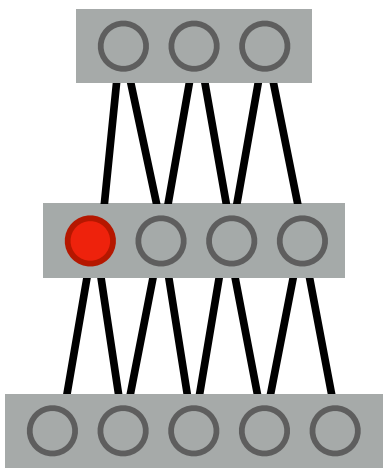


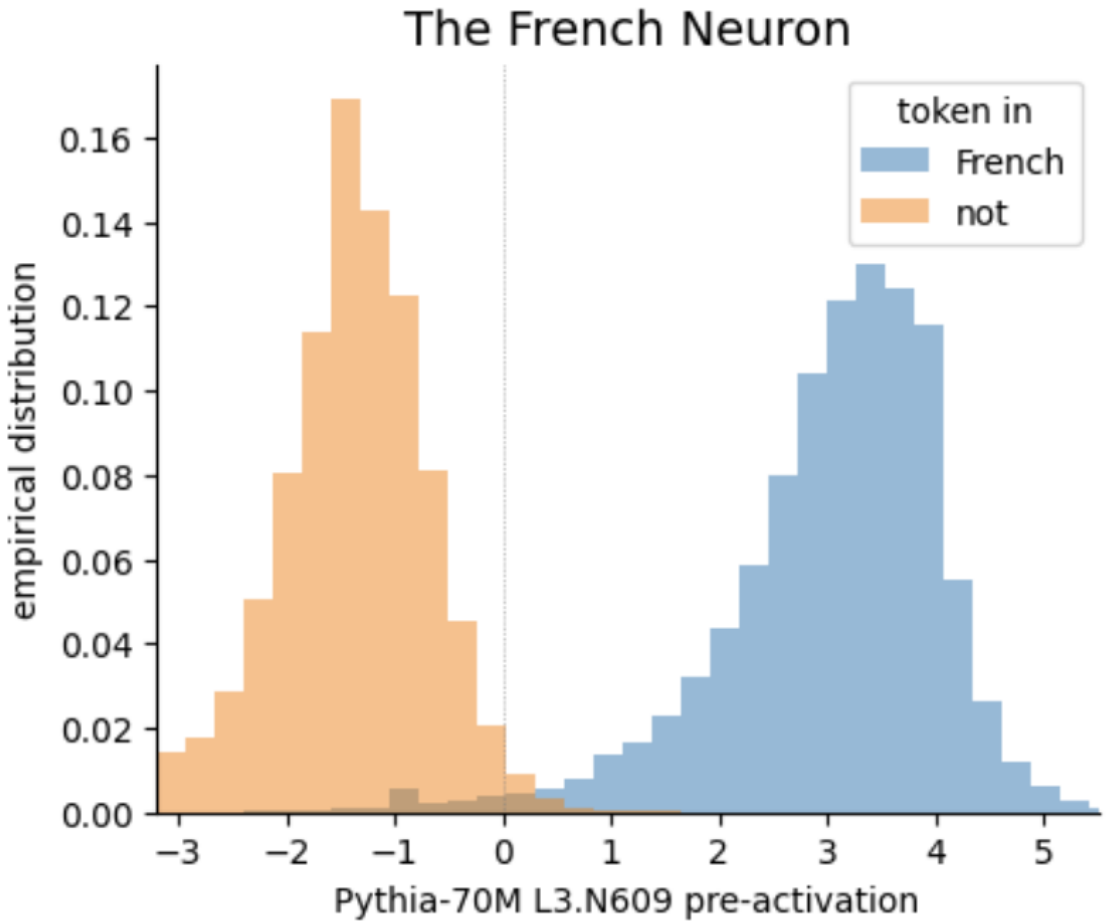


Radford et al. 2017

E.g.,. The French Neuron

- Sometimes the feature of interest is encoded by a single neuron





Comprehesive Probing Study

What you can cram into a single $&!#* vector:
Probing sentence embeddings for linguistic properties

Alexis Conneau
Facebook AI Research
Université Le Mans
aconneau@fb.com

German Kruszewski
Facebook AI Research
germank@fb.com

Guillaume Lample
Facebook AI Research
Sorbonne Universités
glample@fb.com

Loïc Barrault
Université Le Mans
loic.barrault@univ-lemans.fr

Marco Baroni
Facebook AI Research
mbaroni@fb.com

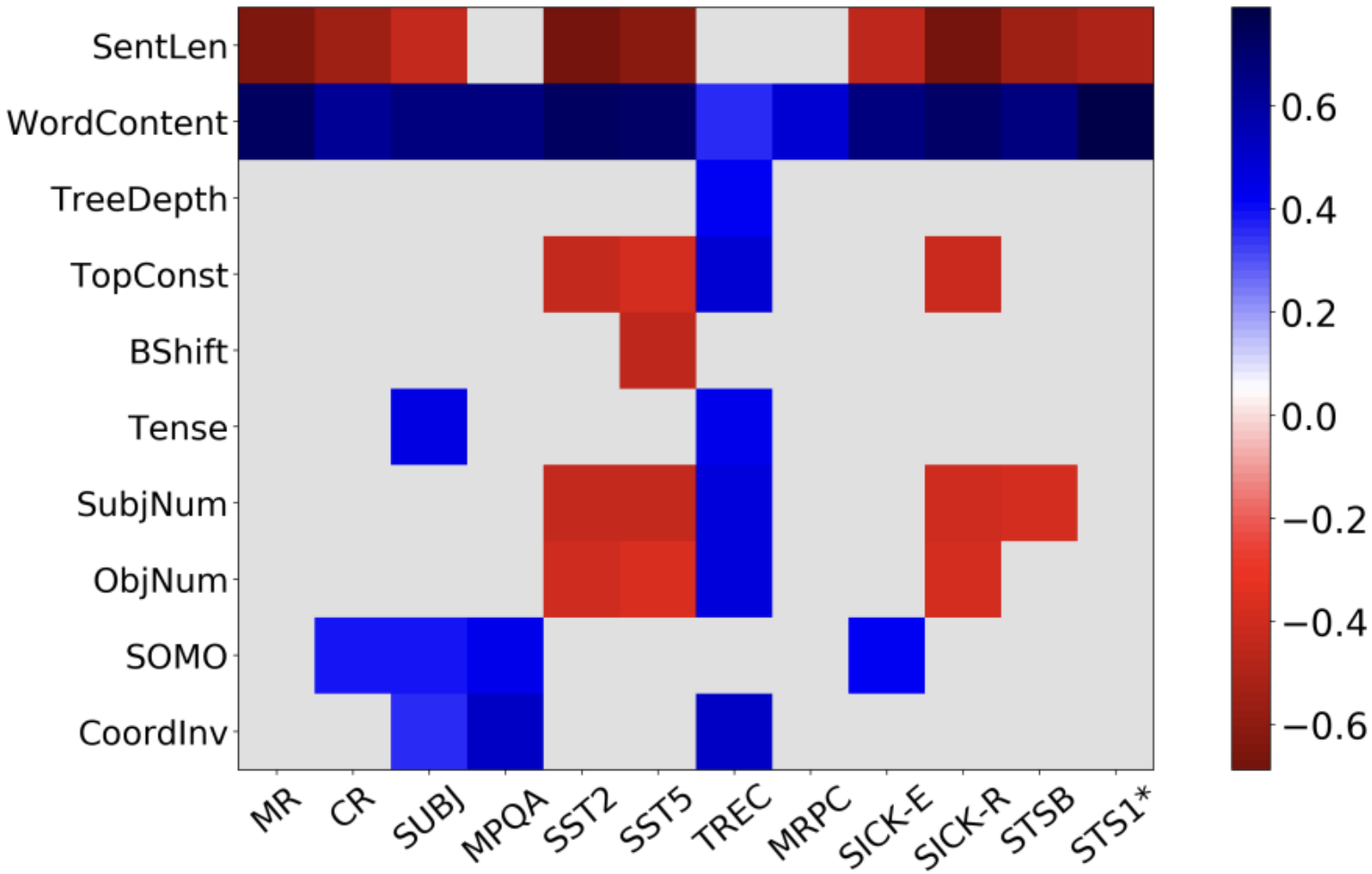
Probing Tasks from Conneau et al., 2018

- **Part-of-Speech:** Identifying the part-of-speech of tokens within a sentence
- **Tree Depth:** Estimating the syntactic complexity of sentences based on their parse tree depths
- **Subject Number:** Determining whether the subject of the sentence is singular or plural
- **Object Number:** Determining whether the object of the sentence is singular or plural
- **Tense:** Identifying the tense of the main verb in a sentence
- **Semantic Odd Man Out:** Detecting the word that does not semantically fit within the sentence.
- **Coordination Inversion:** Determining whether two conjoined phrases in a sentence have been inverted

Task	SentLen	WC	TreeDepth	TopConst	BShift	Tense	SubjNum	ObjNum	SOMO	CoordInv
Baseline representations										
Majority vote	20.0	0.5	17.9	5.0	50.0	50.0	50.0	50.0	50.0	50.0
Hum. Eval.	100	100	84.0	84.0	98.0	85.0	88.0	86.5	81.2	85.0
Length	**100**	0.2	18.1	9.3	50.6	56.5	50.3	50.1	50.2	50.0
NB-uni-tfidf	22.7	**97.8**	24.1	41.9	49.5	77.7	68.9	64.0	38.0	50.5
NB-bi-tfidf	23.0	95.0	24.6	53.0	**63.8**	75.9	69.1	65.4	39.9	**55.7**
BoV-fastText	66.6	91.6	**37.1**	**68.1**	50.8	**89.1**	**82.1**	**79.8**	**54.2**	54.8
BiLSTM-last encoder										
Untrained	36.7	43.8	28.5	76.3	49.8	84.9	84.7	74.7	51.1	64.3
AutoEncoder	**99.3**	23.3	35.6	78.2	62.0	84.3	84.7	82.1	49.9	65.1
NMT En-Fr	83.5	**55.6**	42.4	81.6	62.3	88.1	89.7	89.5	52.0	71.2
NMT En-De	83.8	53.1	42.1	81.8	60.6	88.6	89.3	87.3	51.5	**71.3**
NMT En-Fi	82.4	52.6	40.8	81.3	58.8	88.4	86.8	85.3	52.1	71.0
Seq2Tree	94.0	14.0	**59.6**	**89.4**	**78.6**	**89.9**	**94.4**	**94.7**	49.6	67.8
SkipThought	68.1	35.9	33.5	75.4	60.1	89.1	80.5	77.1	**55.6**	67.7
NLI	75.9	47.3	32.7	70.5	54.5	79.7	79.3	71.3	53.3	66.5
BiLSTM-max encoder										
Untrained	73.3	**88.8**	46.2	71.8	70.6	89.2	85.8	81.9	73.3	68.3
AutoEncoder	**99.1**	17.5	45.5	74.9	71.9	86.4	87.0	83.5	73.4	71.7
NMT En-Fr	80.1	58.3	51.7	81.9	73.7	89.5	90.3	89.1	73.2	75.4
NMT En-De	79.9	56.0	52.3	82.2	72.1	90.5	90.9	89.5	73.4	**76.2**
NMT En-Fi	78.5	58.3	50.9	82.5	71.7	90.0	90.3	88.0	73.2	75.4
Seq2Tree	93.3	10.3	**63.8**	**89.6**	**82.1**	**90.9**	**95.1**	**95.1**	73.2	71.9
SkipThought	66.0	35.7	44.6	72.5	73.8	90.3	85.0	80.6	**73.6**	71.0
NLI	71.7	87.3	41.6	70.5	65.1	86.7	80.7	80.3	62.1	66.8
GatedConvNet encoder										
Untrained	90.3	17.1	30.3	47.5	62.0	78.2	72.2	70.9	61.4	59.6
AutoEncoder	**99.4**	16.8	46.3	75.2	71.9	87.7	88.5	86.5	**73.5**	72.4
NMT En-Fr	84.8	41.3	44.6	77.6	67.9	87.9	88.8	86.6	66.1	72.0
NMT En-De	89.6	49.0	50.5	81.7	72.3	90.4	91.4	89.7	72.8	**75.1**
NMT En-Fi	89.3	**51.5**	49.6	81.8	70.9	90.4	90.9	89.4	72.4	**75.1**
Seq2Tree	96.5	8.7	**62.0**	**88.9**	**83.6**	**91.5**	**94.5**	**94.3**	**73.5**	73.8
SkipThought	79.1	48.4	45.7	79.2	73.4	90.7	86.6	81.7	72.4	72.3
NLI	73.8	29.2	43.2	63.9	70.7	81.3	77.5	74.4	73.3	71.0

Table 2: **Probing task accuracies.** Classification performed by a MLP with sigmoid nonlinearity, taking

Probing Task Correlation with downstream tasks



Conneau et al., 2018

Layerwise trends of probing accuracy

Which layers yield the best performance?

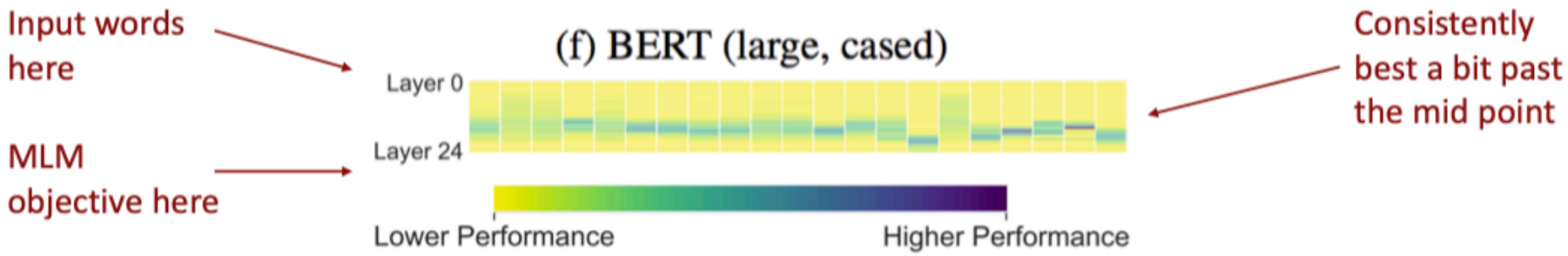


Figure 3: A visualization of layerwise patterns in task performance. Each column represents a probing task, and each row represents a contextualizer layer.

What do we learn from Probing?

Probing is a useful tool but ...

- If probe has high accuracy, what does that mean?
  1. Representation actually encodes information or
  2. Probe classifier solved task by itself

- Probe doesn't work, what does that mean?
  1. Representation lacks the information
  2. Representation encodes information, but probe is not the right function class

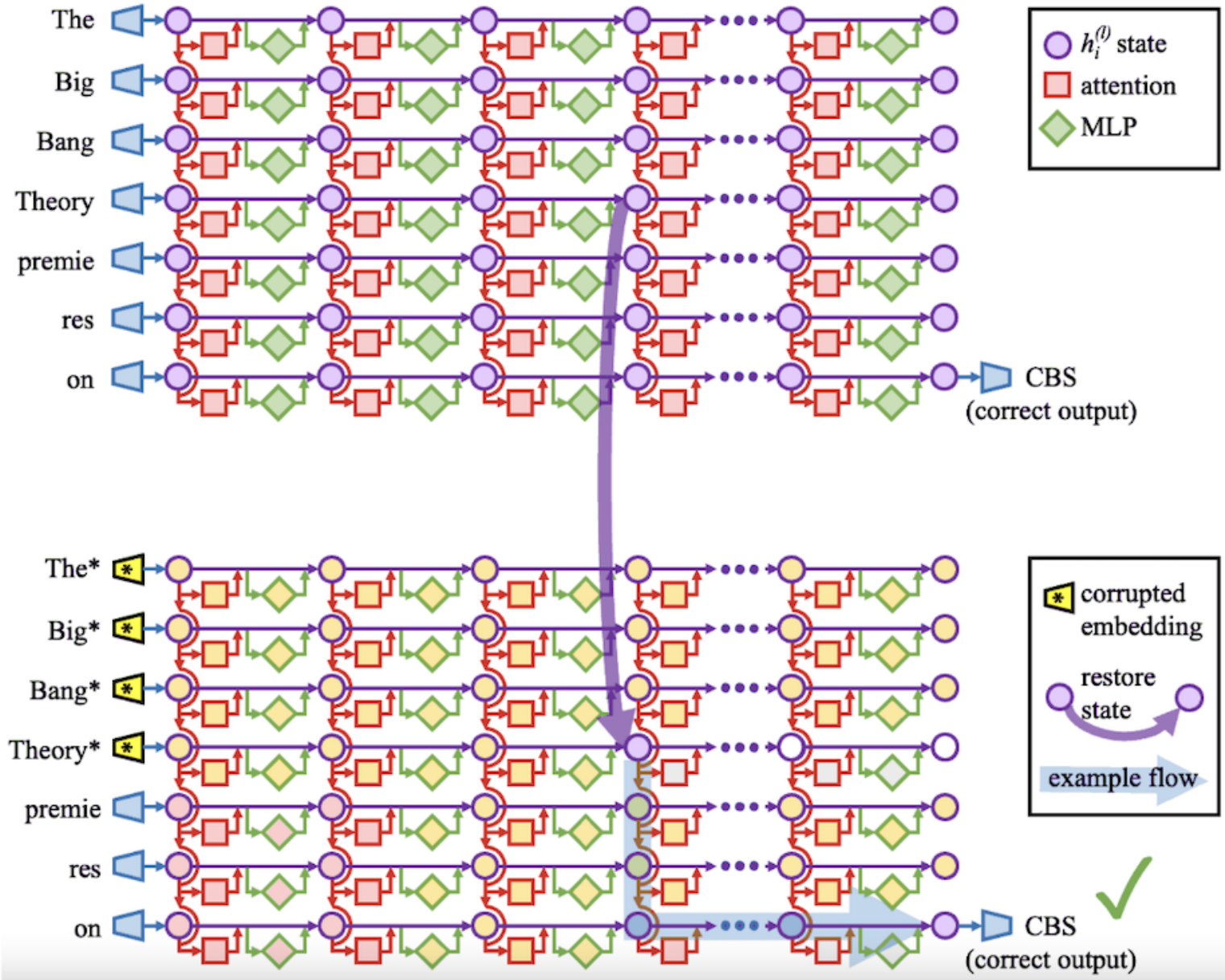
- **Probing Limitation**: Disconnect between the probing classifier g and the original model f
  - ⊙ Did the model use the information discovered by the probe in its decision making?

Belinkov et al., 2021

Analyzing Model Activations on Inputs

Causal Tracing: activation patching

Meng et al (2022)



What do we learn from Causal Tracing via Activation Patching?

- Can be used to understand the role of specific neurons or activations in the model
- But does not tell us human interpretable features

Sparse Autoencoders

Used to discover interpretable features in the model

SPARSE AUTOENCODERS FIND HIGHLY INTERPRETABLE FEATURES IN LANGUAGE MODELS

Hoagy Cunningham[*12], **Aidan Ewart**[*13], **Logan Riggs**[*1], **Robert Huben**, **Lee Sharkey**[4]
[1]EleutherAI, [2]MATS, [3]University of Bristol, [4]Apollo Research
{hoagycunningham, aidanprattewart, logansmith5}@gmail.com

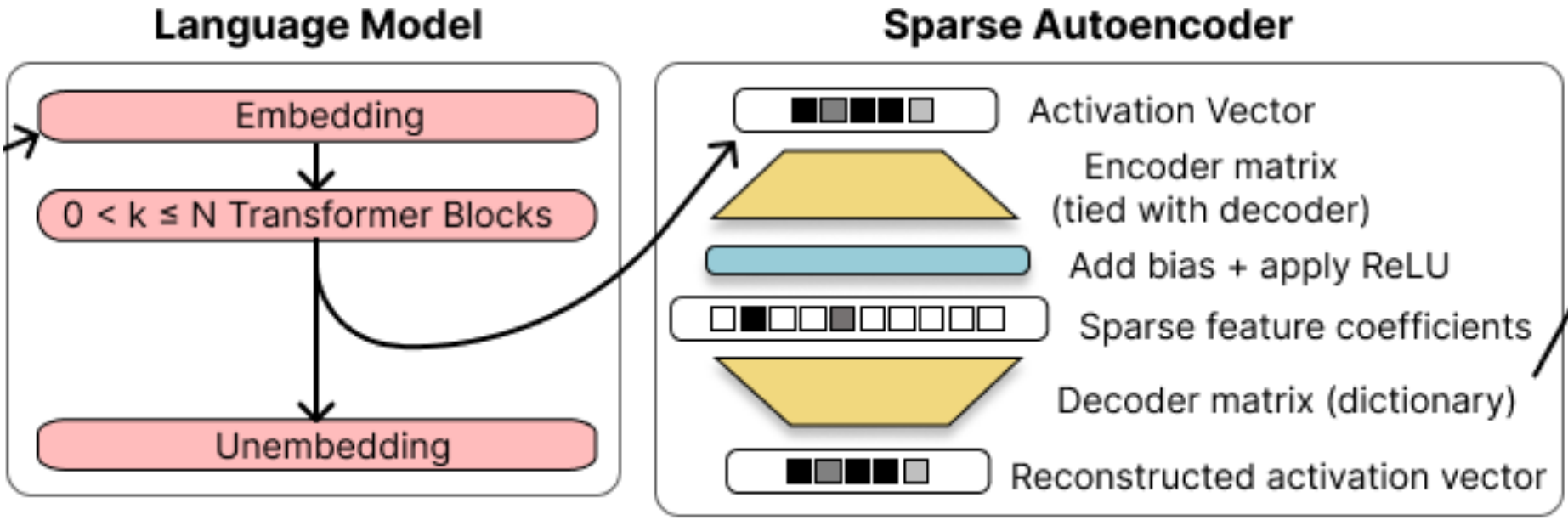
ABSTRACT

One of the roadblocks to a better understanding of neural networks' internals is *polysemanticity*, where neurons appear to activate in multiple, semantically distinct contexts. Polysemanticity prevents us from identifying concise, human-understandable explanations for what neural networks are doing internally. One

Challenge with Interpreting Neurons: Polysemanticity

- **Polysemanticity:** Neurons activate for multiple unrelated features (Olah et al., 2020).
- e.g., a “sentiment neuron" might also activate on orthogonal topics such as “finance" or “law."
  - Law: Activating for phrases like “court ruling" or “legal dispute."
- **Hypothesis on Cause of Polysemanticity:** Neural networks represent more features than they have neurons

From Activations to Sparse Features



a. Sample activations from a language model

b. Learn a feature dictionary using an autoencoder that learns to represent activation vectors as a sparse linear combination of feature vectors.

Features

What do we mean by features?

- A feature is something that activates when it sees a specific concept or ideas
- E.g., The prompt “I like cats", might activate the feature “about cats".

Addressing Superposition with Sparse Autoencoders

Feature Dictionary

Feature	Meaning	Interpretability Score
k-0001	Words ending in "ing"	0.56
k-xxxx	...	...
k-2048	Chemistry terms	0.38

c. Interpret the resulting dictionary features

Sparse Autoencoder: Problem Formulation

- $\{\mathbf{x}_i\}_{i=1}^{n_{\text{vec}}} \subset \mathbb{R}^d$ are activations from an LLM
- Unknown **ground truth features:** $\{\mathbf{f}_j\}_{j=1}^{n_{\text{ft}}} \subset \mathbb{R}^d$
- Assume:

$$\mathbf{x}_i = \sum_j a_{i,j} \mathbf{f}_j$$

  where $\mathbf{a}_i = \left[a_{i,1}, a_{i,2}, \ldots, a_{i,n_{\text{gt}}}\right]^\top$ is a sparse vector, meaning most $a_{i,j}$ are 0 , and only a few scalar coefficients are nonzero

- Learning objective: Learning $\{\mathbf{f}_j\}$ and $\{\mathbf{a}_i\}$ from $\{\mathbf{x}_i\}$.

Huben et al., ICLR 2024

Sparse Representations

- **Given Equation:**

$$\mathbf{x}_i = \sum_j a_{i,j} \mathbf{f}_j$$

  - $\mathbf{x}_i$: Data vector (e.g., activations from an LLM)
  - $\mathbf{f}_j$: feature vectors (shared across all $\mathbf{x}_i$)
  - $a_{i,j}$: Coefficients determining how much each $\mathbf{f}_j$ contributes to $\mathbf{x}_i$
- **Role of $\mathbf{a}_i$:**
  - Each $\mathbf{x}_i$ has a unique sparse vector $\mathbf{a}_i$, containing the coefficients $\{a_{i,j}\}$
  - The sparse vector $\mathbf{a}_i$ encodes how $\mathbf{x}_i$ is reconstructed as a weighted combination of the feature vectors $\mathbf{f}_j$.

Sparse Representations

- $\mathbf{f}_j$: Shared feature vectors (e.g., syntax, semantics, etc.).
- $\mathbf{a}_i$: Unique sparse vector for $\mathbf{x}_i$, selecting which $\mathbf{f}_j$ vectors contribute to $\mathbf{x}_i$.

For $\mathbf{x}_1$:

$$\mathbf{a}_1 = [0, 2.5, 0, 0.8, 0]$$

This means $\mathbf{x}_1$ is reconstructed using:

$$\mathbf{x}_1 = 2.5 \cdot \mathbf{f}_2 + 0.8 \cdot \mathbf{f}_4$$

For $\mathbf{x}_2$:

$$\mathbf{a}_2 = [1.2, 0, 0, 0, 0.5]$$

This means $\mathbf{x}_2$ is reconstructed using:

$$\mathbf{x}_2 = 1.2 \cdot \mathbf{f}_1 + 0.5 \cdot \mathbf{f}_5$$

Autoencoder Architecture

Train a single-layer **autoencoder** with a sparsity penalty to learn feature dictionaries

Encoder:

$$\mathbf{c} = \mathrm{ReLU}(M\mathbf{x} + \mathbf{b})$$

Decoder:

$$\hat{\mathbf{x}} = M^T\mathbf{c} = \sum_{i=0}^{d_{\text{hid}}-1} c_i \mathbf{f}_i$$

- $M \in \mathbb{R}^{d_{\text{hid}} \times d_{\text{in}}}$: Learned weight matrix, normalized row-wise
- **Tied weights:** Decoder weights are the transpose of encoder weights (M^T)

Feature Dictionary

Encoder:

$$\mathbf{c} = \mathrm{ReLU}(M\mathbf{x} + \mathbf{b})$$

Decoder:

$$\hat{\mathbf{x}} = M^T\mathbf{c} = \sum_{i=0}^{d_{\text{hid}}-1} c_i \mathbf{f}_i$$

- Rows of M, denoted $\{\mathbf{f}_i\}$, represent the learned features
- Hidden layer activations $\mathbf{c}$ provide sparse coefficients for reconstructing $\mathbf{x}$

Minimize:

$$\mathcal{L}(\mathbf{x}) = \underbrace{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}_{\text{Reconstruction loss}} + \underbrace{\alpha\|\mathbf{c}\|_1}_{\text{Sparsity loss}}$$

- α: Hyperparameter controlling sparsity.

Features

Explain what patterns in text that fire strongly on features

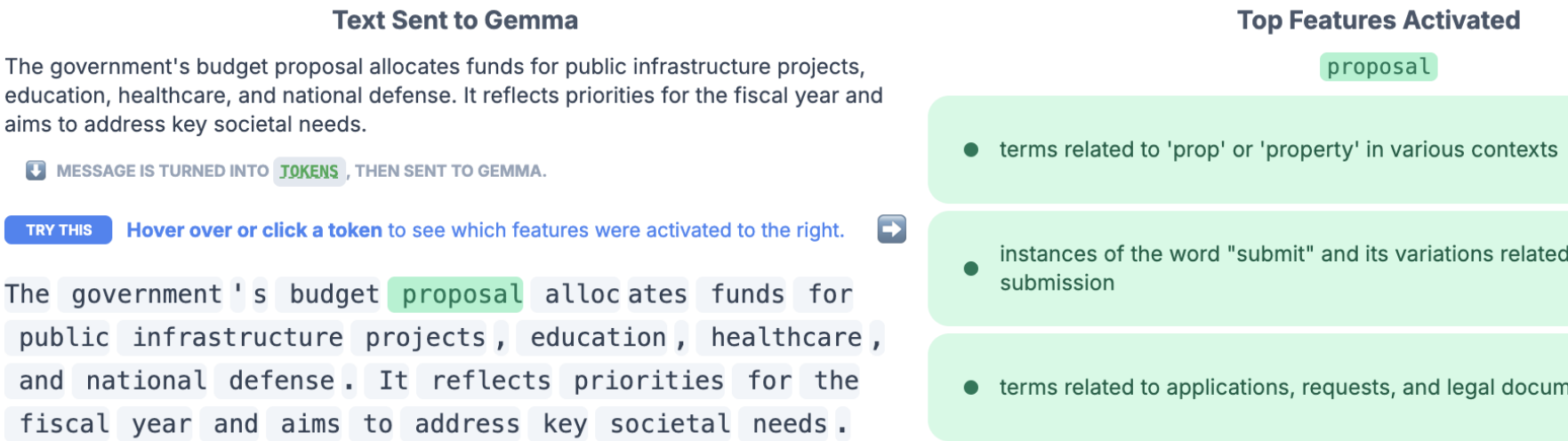
Feature	Description (Generated by GPT-4)	Interpretability Score
1-0000	parts of individual names, especially last names.	0.33
1-0001	actions performed by a subject or object.	-0.11
1-0002	instances of the letter 'W' and words beginning with 'w'.	0.55
1-0003	the number '5' and also records moderate to low activation for personal names and some nouns.	0.57
1-0004	legal terms and court case references.	0.19

Table 1: Results of autointerpretation on the first five features found in the layer 1 residual stream. Autointerpretation produces a description of what the feature means and a score for how well that description predicts other activations.

- Found that the learned features correspond to human-understandable concepts
- They are more interpretable than neuron activations (less polysemantic)

Gemma Scope Features

Gemma Scope Features



Summary of Sparse Autoencoders

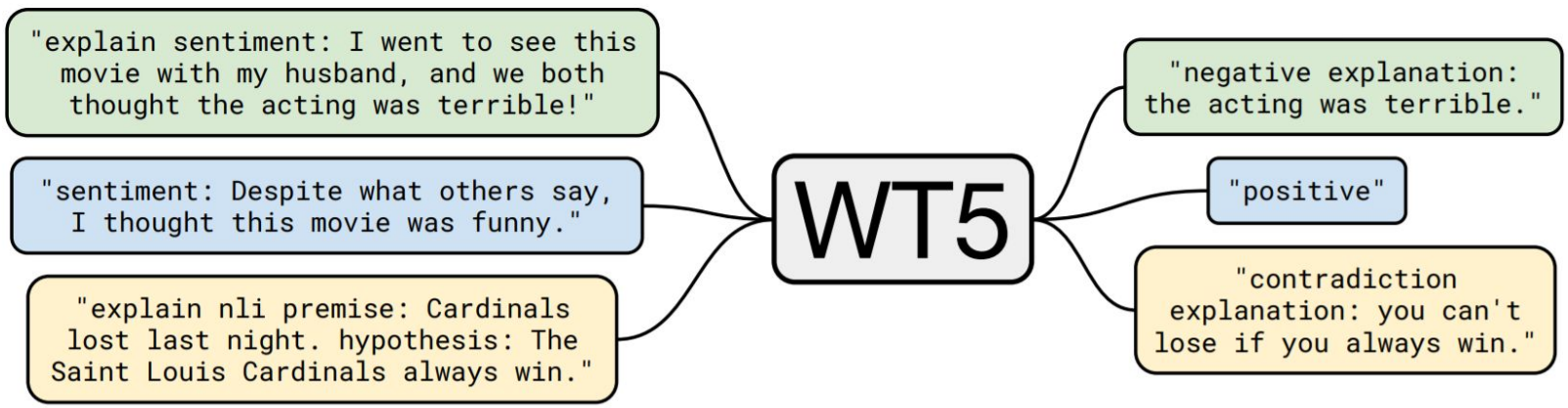
- Sparse autoencoders: **unsupervised method** for disentangling features in language models from superposition.
- Require only unlabelled activations and less compute than original model training.
- Learned dictionary features are:
  - **Monosemantic:** Pinpoints features responsible for specific behaviors

Naitive Interpretability

Can we design models that are more interpretable by design?

Baking interpretability into the model

Q: Can we design models that are more interpretable by design?



[Lei et al. 2016, Narang et al. 2020, DeYoung et al. 2020, ...]

A: Yes, but not much work on this.

Dataset Artifacts

Many datasets are easy to solve with big models. What does that tell us about the models?

Are **humans** robust to noise in their input?

"Aoccdrnig to a rscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae."

Are **models** robust to noise in their input?

- Noise of various kinds is an inevitable part of the inputs to NLP systems.
- How do models trained on (relatively) clean text perform when typo-like noise is added?
- Belinkov and Bisk, 2018 performed a study on popular machine translation models.

BLEU scores are high on in-domain clean text

Character-swaps like we just saw break the model!

(More) natural typo noise also breaks the models.

			Synthetic				
		Vanilla	Swap	Mid	Rand	Key	Nat
French	charCNN	42.54	10.52	9.71	1.71	8.26	17.42
German	charCNN	34.79	9.25	8.37	1.02	6.40	14.02
	char2char	29.97	5.68	5.46	0.28	2.96	12.68
	Nematus	34.22	3.39	5.16	0.29	0.61	10.68
Czech	charCNN	25.99	6.56	6.67	1.50	7.13	10.20
	char2char	25.71	3.90	4.24	0.25	2.88	11.42
	Nematus	29.65	2.94	4.09	0.66	1.41	11.88

Explanation by input reduction

What is the smallest part of the input I could keep and still get the same answer?

Passage: In 1899, John Jacob Astor IV invested $100,000 for Tesla to further develop and produce a new lighting system. Instead, Tesla used the money to fund his Colorado Springs experiments. [prediction]

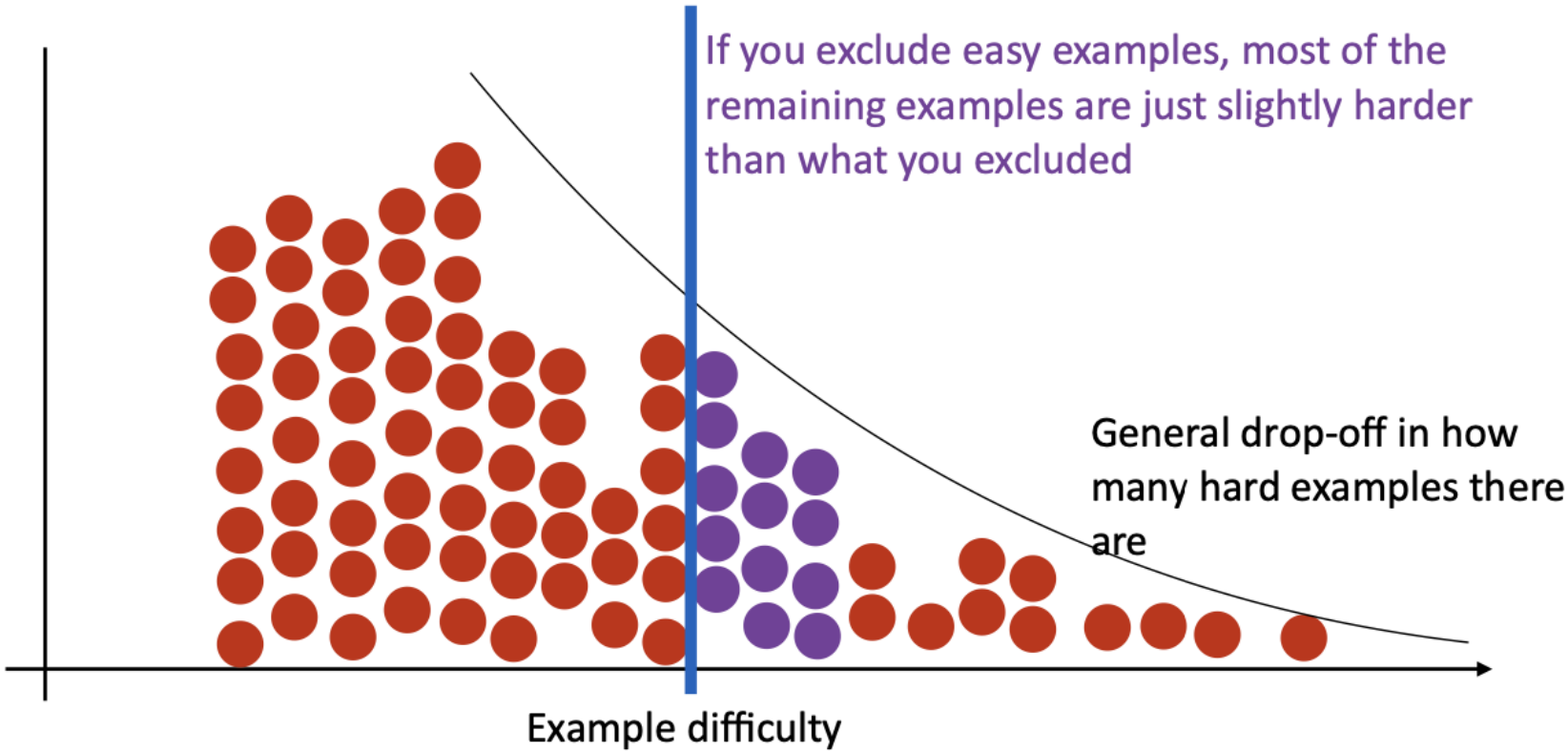
Original Question: What did Tesla spend Astor's money on ?

Reduced Question did

In this example, the model had confidence 0.78 for the original question, and the same answer at confidence **0.91** for the reduced question!

Feng et al., 2018

One problem: datasets contain a lot easy examples

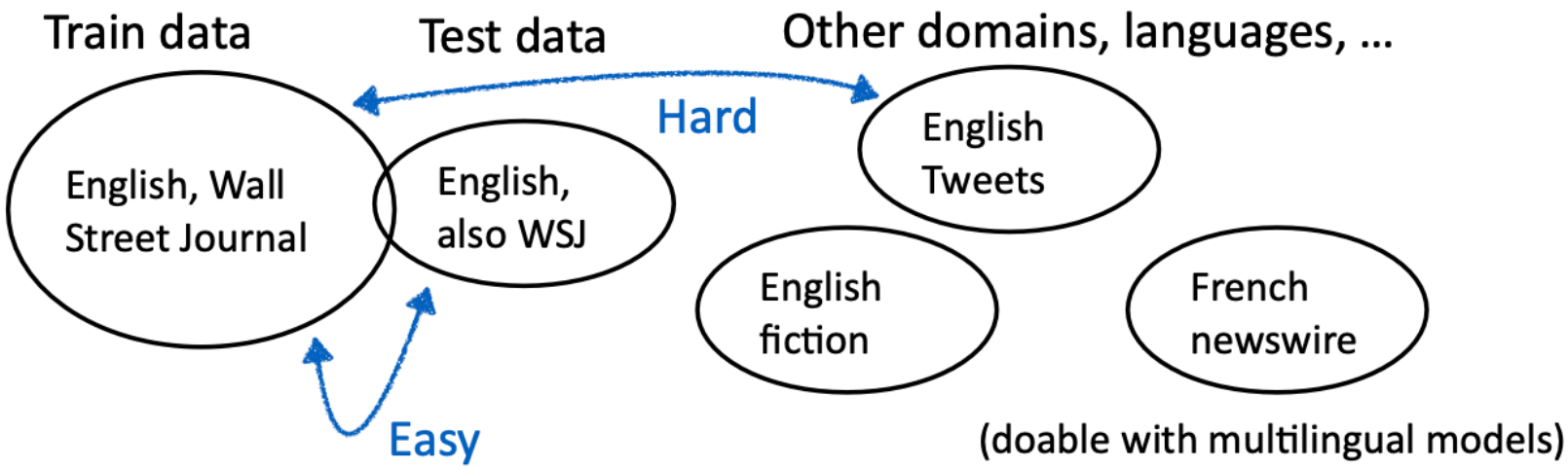


Dataset Artifacts: Evaluation Under Distribution Shifts

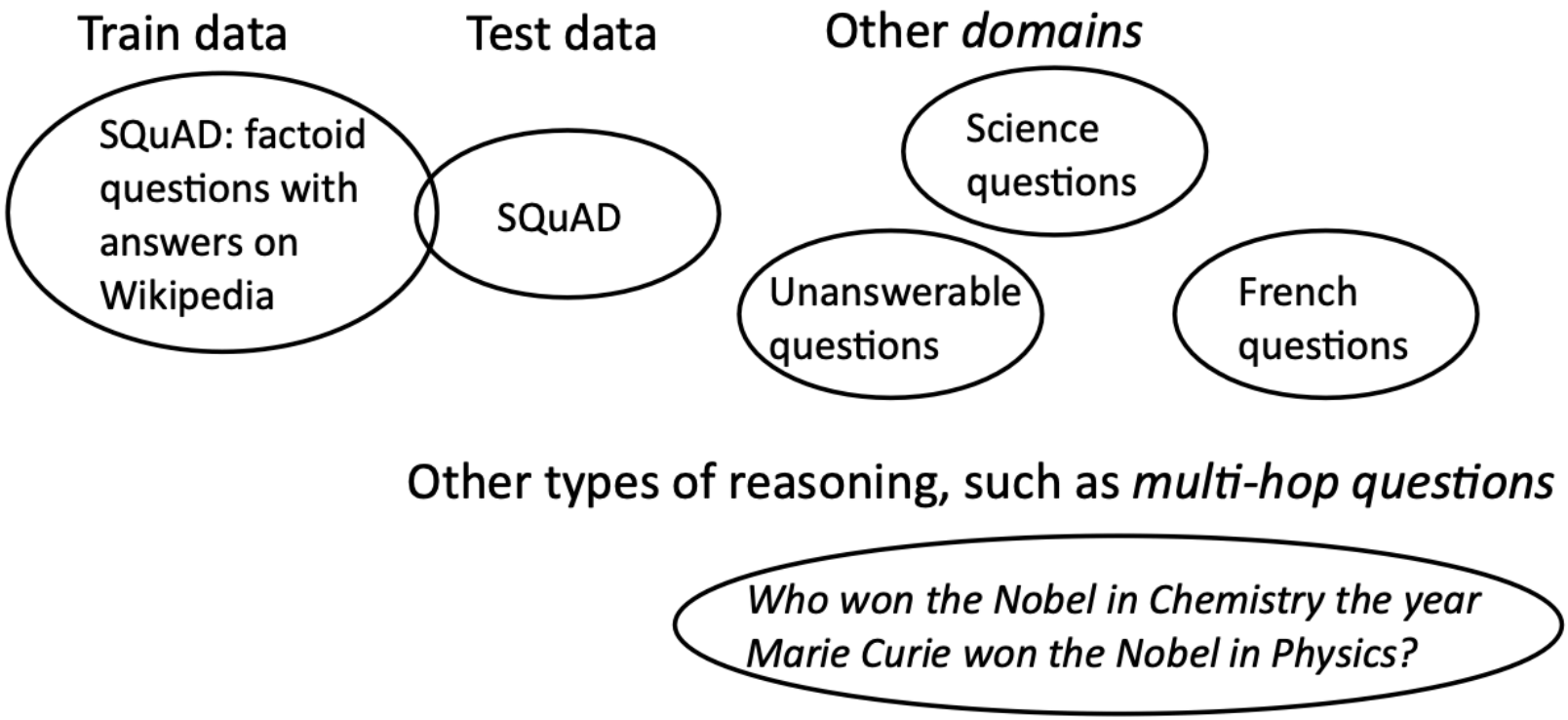
Generalization

- If a model does well on train but poorly on test data, it **doesn't generalize**
- A model can do well on its test data and still fail to generalize **out of distribution** - arguably an even more important notion

Generalization: Part-of-Speech Tagging (POS)



Generalization: Question Answering (QA)



Generalization

- Just doing well on a single test set is not that useful
- We want POS taggers, QA systems, and more that can **generalize to new settings** so we can deploy them in practice
- Sometimes, you can get very good test performance but the model generalizes very poorly. How does this happen?
  - **Spurious correlations** due to artifacts in the training data
  - **Reasoning shortcuts** that don't generalize
  - Models can do well on SQuaD, without looking at the question, passage-only baselines do well

Models tend to learn the dataset, not the task!

Across a wide range of tasks, high model accuracy on the in-domain test set does not imply the model will also do well on other, "reasonable" out-of-domain examples.

One way to think about this: models seem to be learning the dataset (like SQuaD) not the task (like how humans can perform question answering).

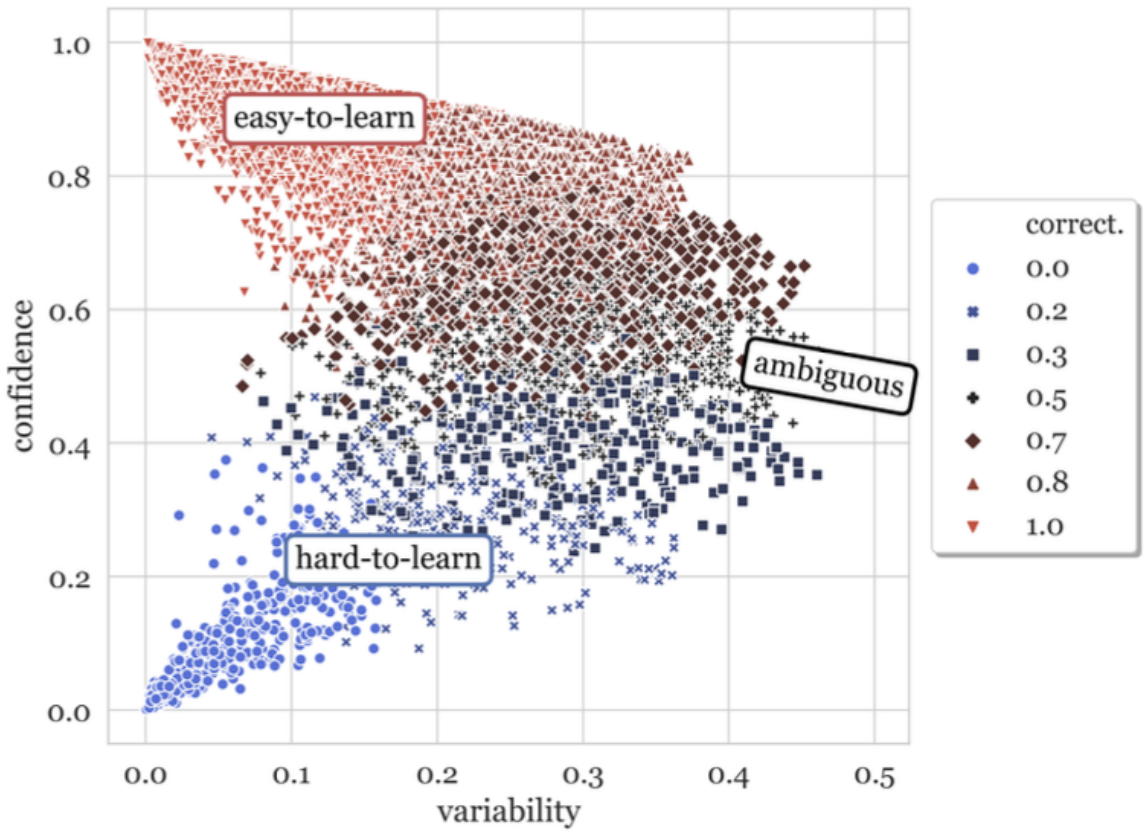
Solutions to Dataset Artifacts

Dataset Cartography

- What happens with each particular example during training epochs
- Spurious correlations are easy to learn: a model should learn these early and always get them right
- Imagine a **very challenging example**
  - Model prediction may change a lot as it learns this example, may be variable in its predictions
- Imagine a **mislabeled example**
  - Probably just always wrong unless it gets overfit

Data Maps

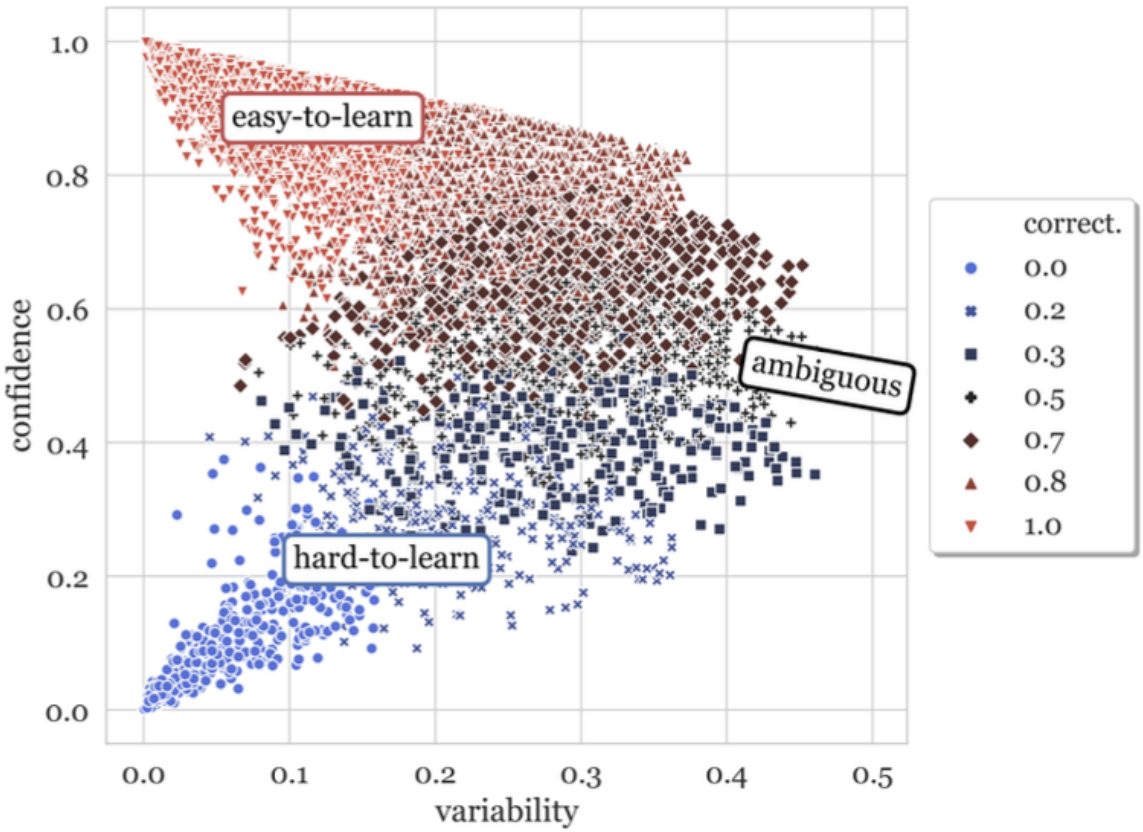
- **Confidence**: probability of correct label
- **Variability**: standard deviation in probability of the correct label across epochs
- Ambiguous examples: possibly learnable (model knows it sometimes but not other times), but hard!



Swayamdipta et al. (202)

Data Maps

- What to do with them?
- Training on hard-to-learn or ambiguous examples leads to better performance out-of-domain



Swayamdipta et al. (202)

Debiasing

- Other ways to identify easy examples other than data maps
- Train some kind of a weak model and discount examples that it fits easily
  - If the weak model is a good predictor, the example is easy and should be discounted in contribution to the loss

one-hot label vector

log probability of each label

$$\mathcal{L}(\theta_d) = -(1 - p_b^{(i,c)})y^{(i)} \cdot \log p_d$$

probability under a copy of the model trained for a few epochs on a small subset of data (bad model)

Utama et al. (2020)

Datasets Takeaways

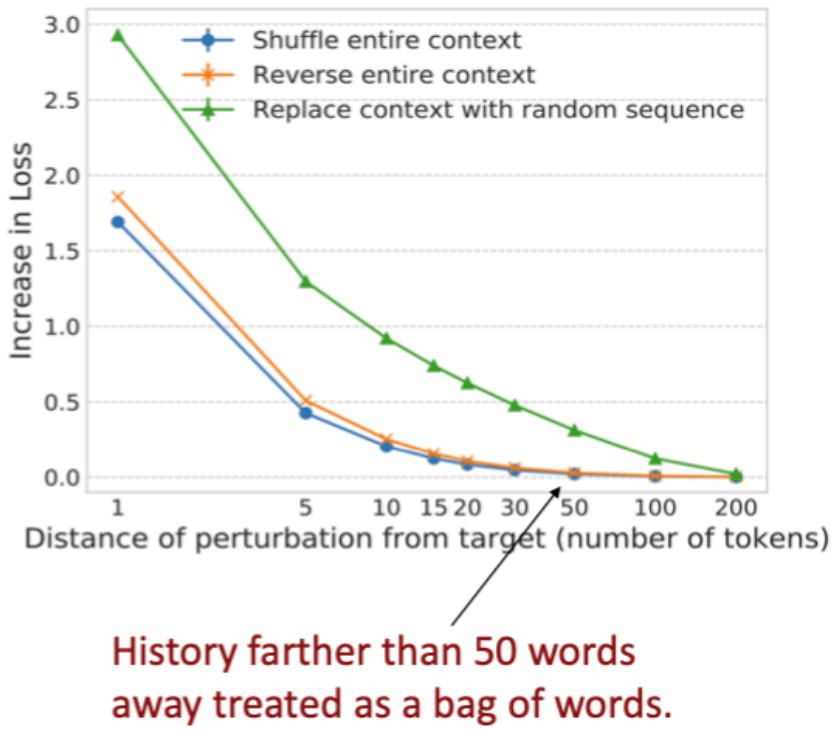
- Strong neural models trained on "tough" datasets may fail to generalize because they learn annotation artifacts
- By reweighting data or changing the training paradigm, you can learn a model that generalizes better
- Most gains will show up out-of-domain
- As more "generalist" LLMs are learned, this problem goes away...but there's always a tradeoff when you want to fine-tune them for certain tasks

Other Considerations

Reasoning when the context is long

Input influence: does my model really use long-distance context?

- We motivated LSTMs language models through their theoretical ability to use long-distance context to make predictions. But how long really is the long short-term memory?
- Khandelwal et al., 2018's idea: shuffle or remove all contexts farther than k words away for multiple values of k and see at which k the model's predictions start to get worse!

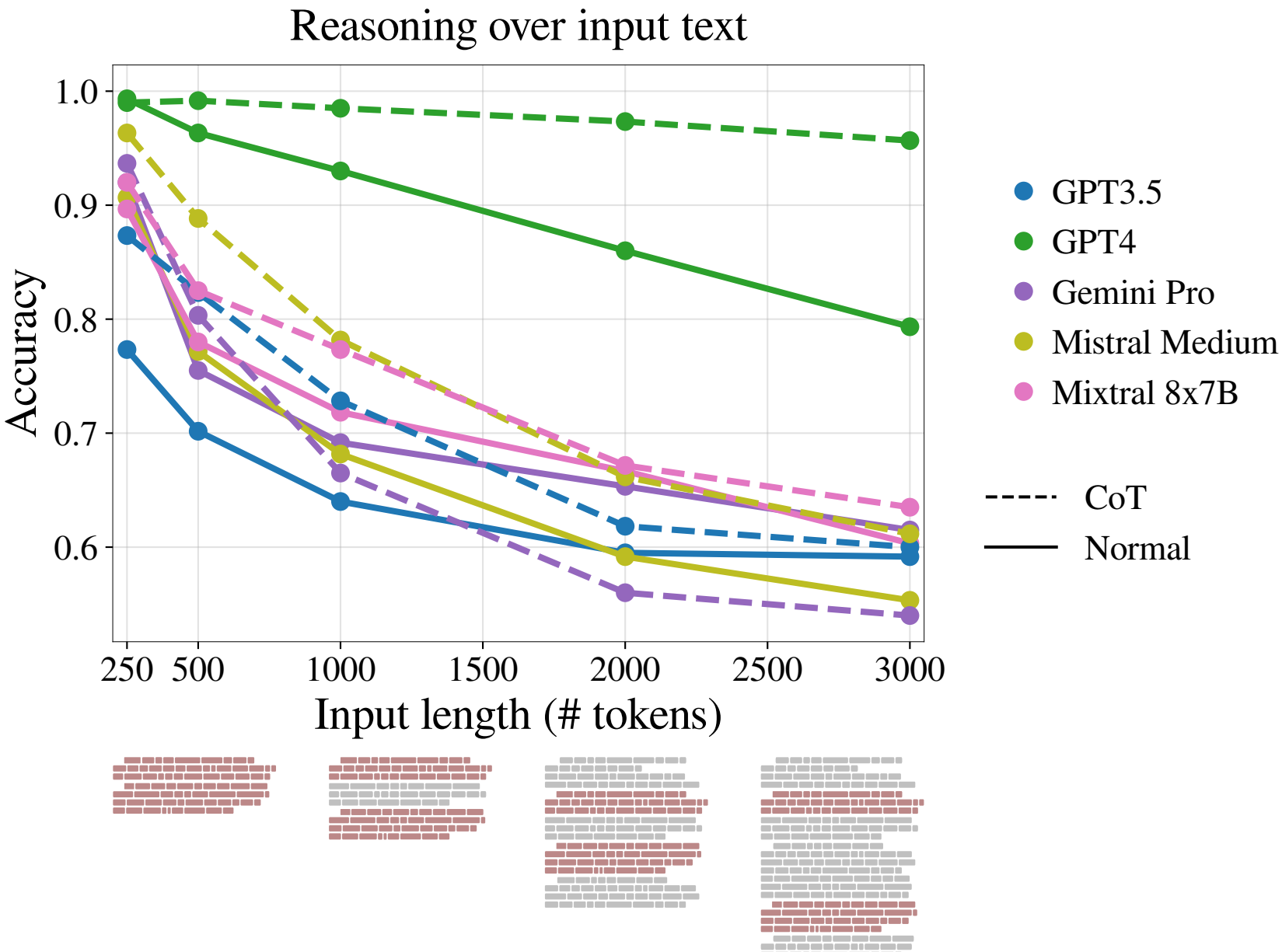


Same Task, More Tokens: the Impact of Input Length on the Reasoning Performance of Large Language Models

Mosh Levy[*1] **Alon Jacoby**[*1] **Yoav Goldberg**[1,2]
[1]Bar-Ilan University [2]Allen Institute for AI
{moshe0110, alonj4}@gmail.com

Reasoning Performance Drops as Input Length Increases

Levy et al. (ACL 2024)



Controlling for Length

Ensuring models reason over the input

- Each data sample should **contain** several relevant text spans that are necessary & sufficient to correctly solve the task
- All relevant spans must be consulted **jointly** to reach a successful solution
- The question and supporting relevant spans should consist of novel facts not seen in training data

Levy et al. (ACL 2024)

Isolating the Length Effect: Task 1: Monotone relations (MonoRel)

fictional names; The relations are transitive and monotone in nature; names are randomly generated with Faker python library

MonoRel Example:

Julie Baker is younger than Julian Barton.

This is a fact that remains constant, unchanging like the northern star. It's a truth that is as clear as day that she ...

Samantha Arnold is younger than Julie Baker.

It means that Samantha Arnold has experienced fewer birthdays than Julie Baker. ...

Is Samantha Arnold younger than Julian Barton?

Task 2: People In Rooms (PIR)

"padding" is done by an LLM

PIR Example:

John's living room is marble-floored, a reality that is as intrinsic to the building as its very foundations. The moment ...

Ethan Washington is in John's living room, a fact that has become as much a part of the place as the walls and the ceiling. The truth that Ethan Washington is in John's living ...

Is Ethan Washington in a marble-floored room?

Levy et al. (ACL 2024)

Task 3: Ruletaker

Simplified Ruletaker Example:

Facts:

Erin is furry. Erin is known for his furriness. He has a lot of fur and ...

Erin is good. Erin was always known for how good he is. His goodness appears on all matters of life ...

Rule:If X is big and X is good then X is tall.

Question: can the statement "Erin is tall" be derived from the rule and the facts?

Levy et al. (ACL 2024)

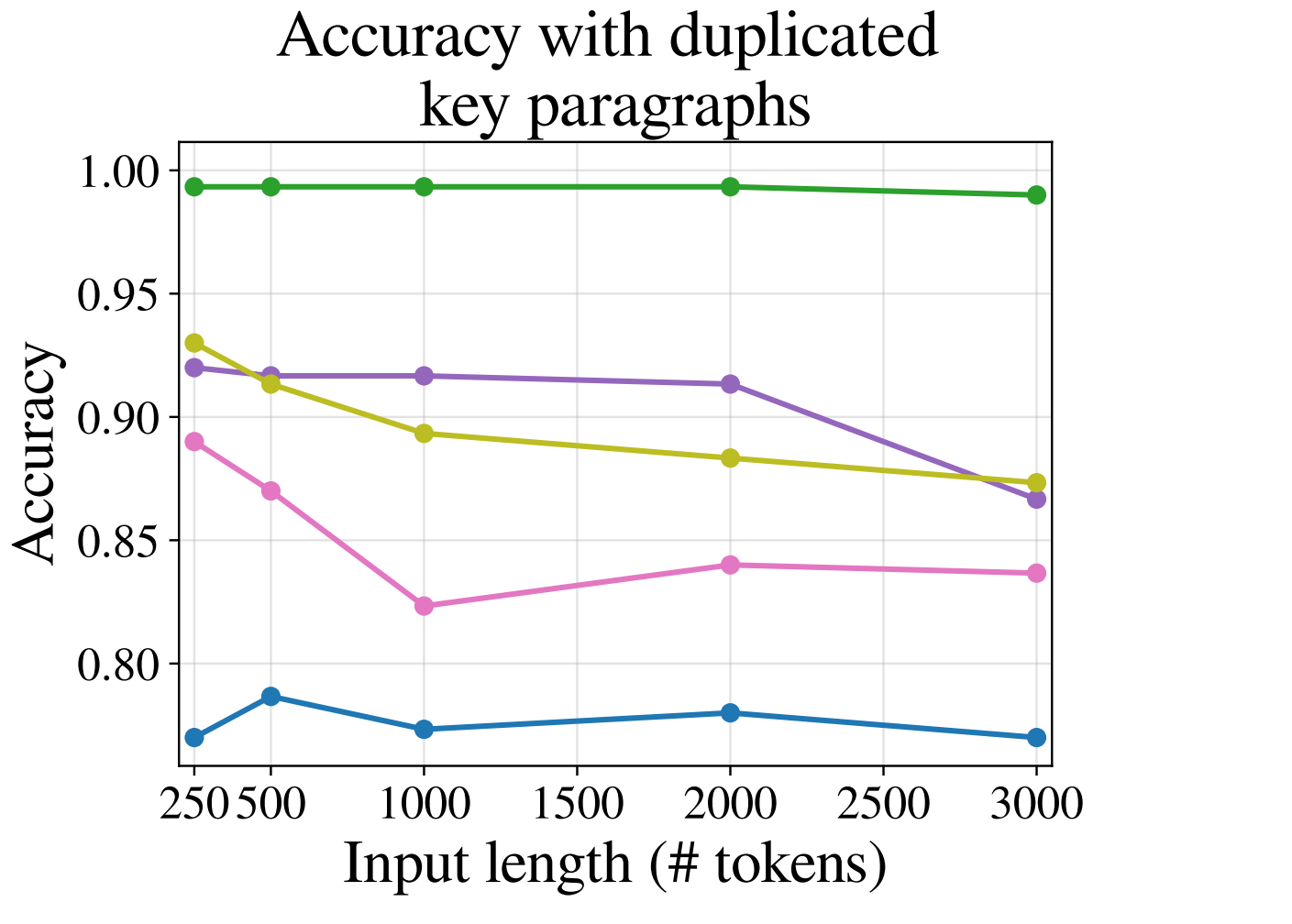
Minimum Length Accuracy

Model	Prompt	MonoRel	PIR	Ruletaker*
GPT3.5	Direct	0.77	0.81	0.74
	CoT	0.86	0.88	0.88
GPT4	Direct	1.00	1.00	0.98
	CoT	1.00	1.00	0.97
Gemini Pro	Direct	0.84	1.00	0.92
	CoT	0.88	0.96	0.97
Mistral 70B	Direct	0.99	1.00	0.73
	CoT	1.00	1.00	0.89
Mixtral 8x7B	Direct	0.92	0.97	0.80
	CoT	0.86	0.97	0.93

Table 1: **Minimal length accuracy.** The evaluated models have high accuracy on the tasks in our dataset when evaluated on the minimal text (250 tokens). CoT improve performance across almost all tasks and models.

No irrelevant paragraphs: Duplicate padding

GPT 3.5/4 are less affected by duplicated padding



Placement of the Relevant Paragraphs

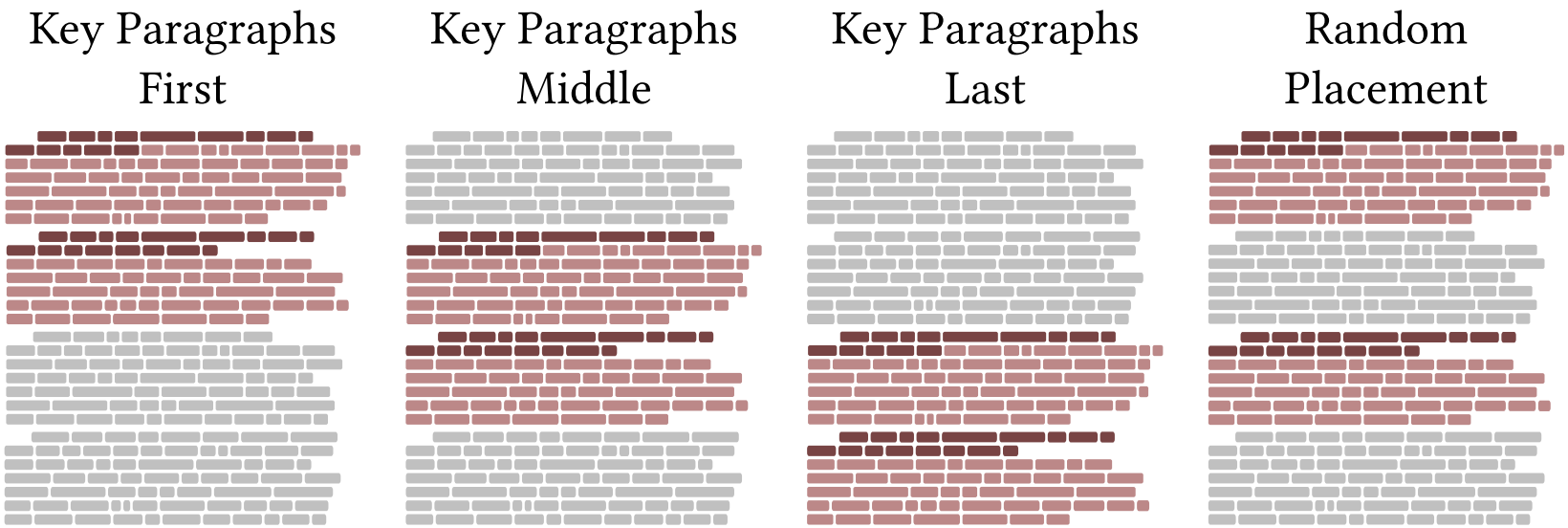
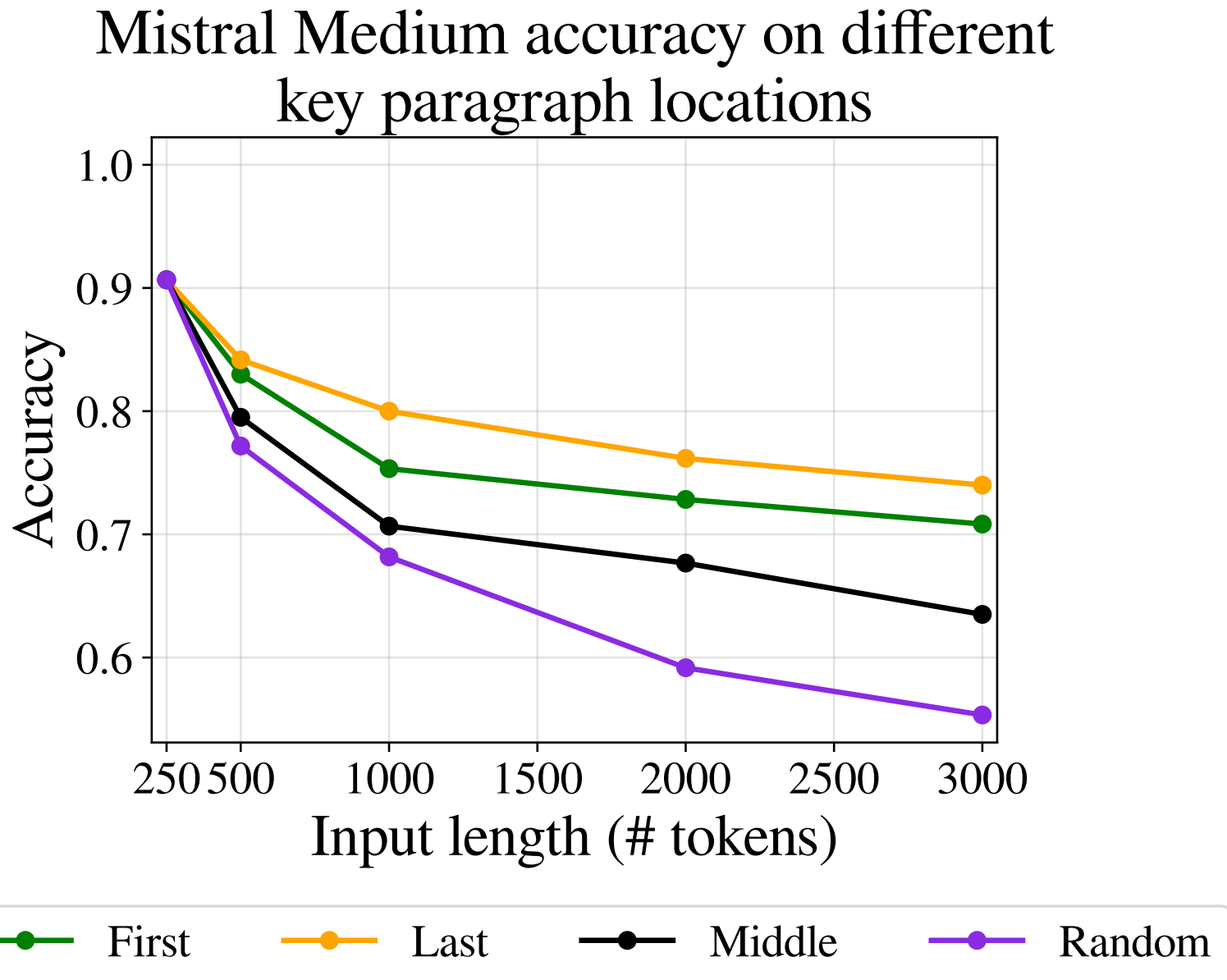


Figure 2: **Inputs construction.** Key sentences (dark red), are expanded to key paragraphs (light red) which are dispersed in controlled locations among padding text (grey) which is irrelevant to the task.

Levy et al. (ACL 2024)

Placement of the Relevant Paragraphs doesn't matter



Takeaways from LLM Length Experiments

- LLMs are sensitive to input length
- Found persistent negative effects of increased length on reasoning performance, regardless of input adjustments
- LLMs are not reasoning over the input

That's all for today