Understanding in-context learning

CS685 Spring 2023

Advanced Natural Language Processing

Mohit Iyyer

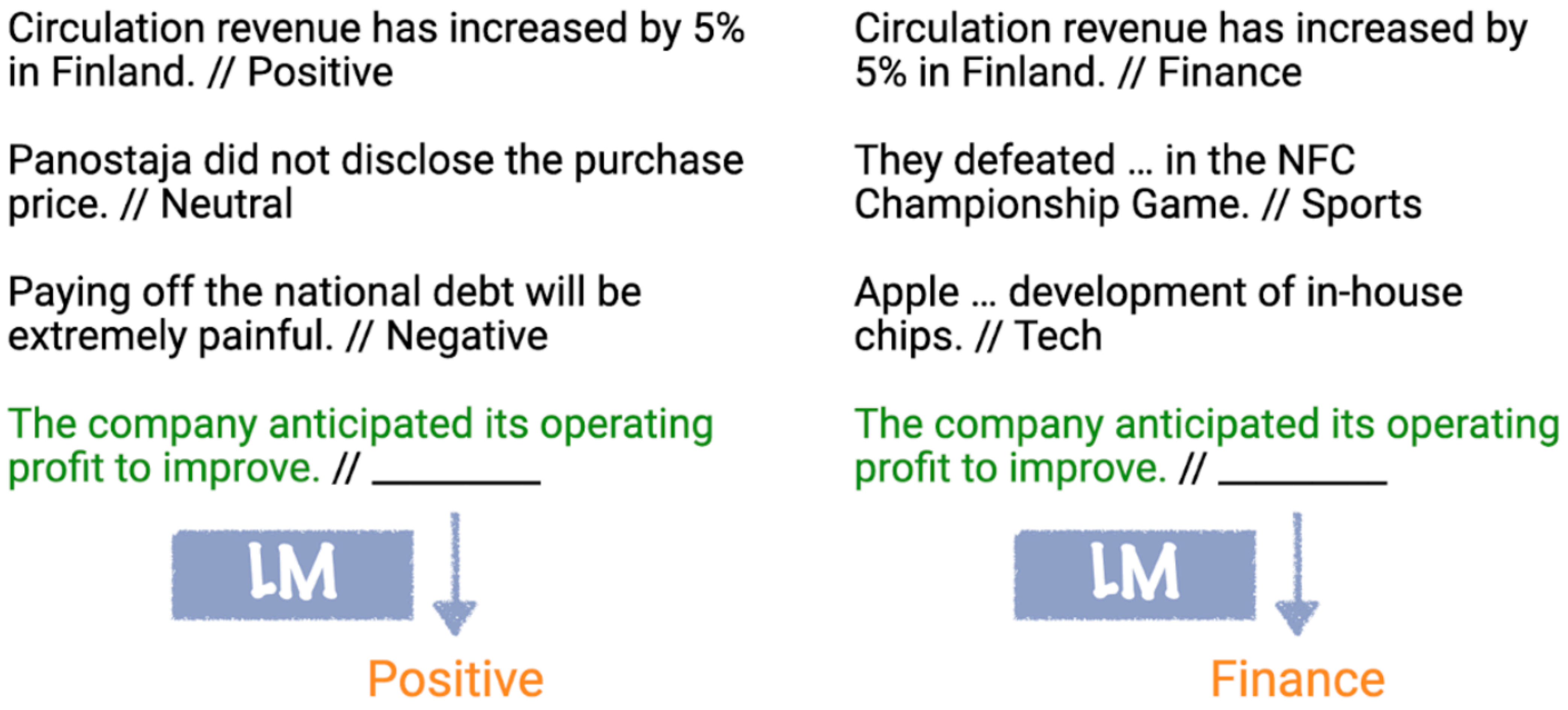
College of Information and Computer Sciences

University of Massachusetts Amherst

In-context learning:
LLMs can solve novel downstream tasks by just conditioning on a few demonstrations of the task in its prefix

In-context learning:

LLMs can solve novel downstream tasks by just conditioning on a few demonstrations of the task in its prefix



Circulation revenue has increased by 5% in Finland. // Positive

Panostaja did not disclose the purchase price. // Neutral

Paying off the national debt will be extremely painful. // Negative

The company anticipated its operating profit to improve. // ________

Circulation revenue has increased by 5% in Finland. // Finance

They defeated … in the NFC Championship Game. // Sports

Apple … development of in-house chips. // Tech

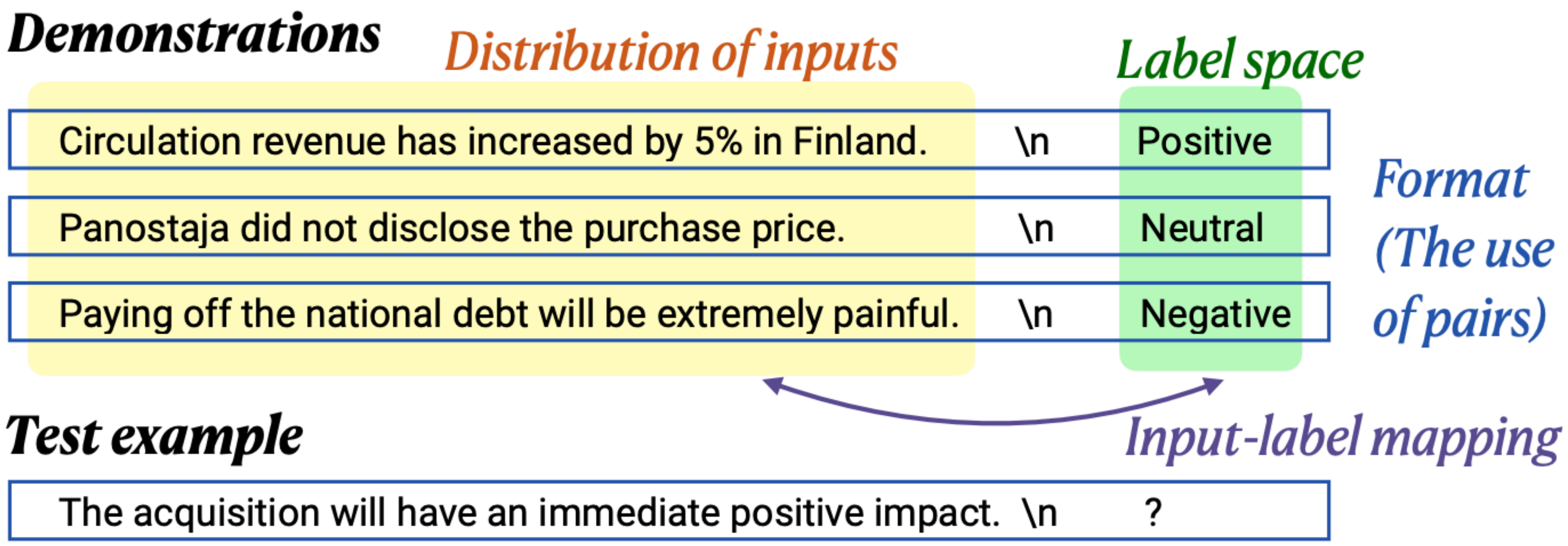
The company anticipated its operating profit to improve. // ________

https://ai.stanford.edu/blog/understanding-incontext/

How does this work?

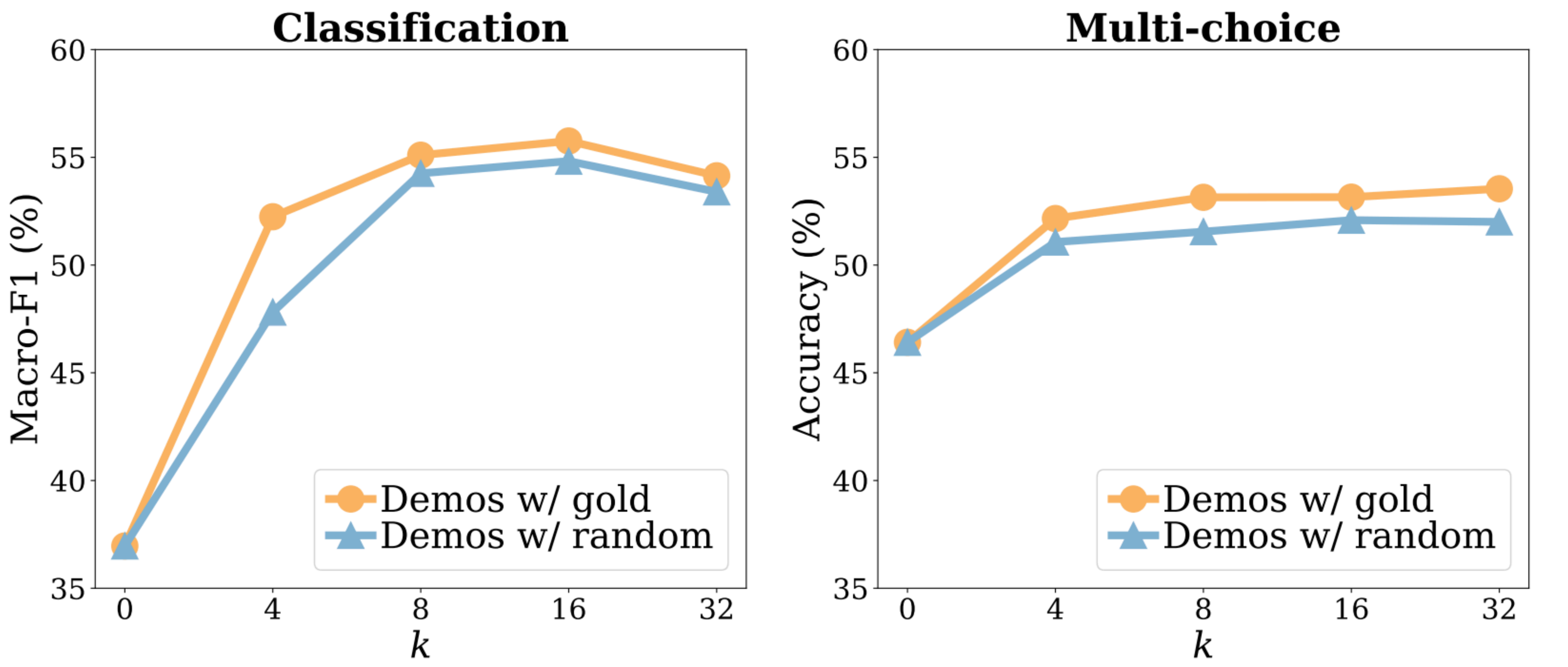
- Are “novel” tasks really novel? Or are they buried somewhere within the trillions of words in the Common Crawl?
- If not, how can the model “learn” to solve a new task without any training (i.e., gradient updates to optimize model performance on the new task)?

What's in a demonstration?


Demonstrations
Distribution of inputs
Label space
Circulation revenue has increased by 5% in Finland. \n Positive
Panostaja did not disclose the purchase price. \n Neutral
Paying off the national debt will be extremely painful. \n Negative
Format (The use of pairs)
Input-label mapping
Test example
The acquisition will have an immediate positive impact. \n ?

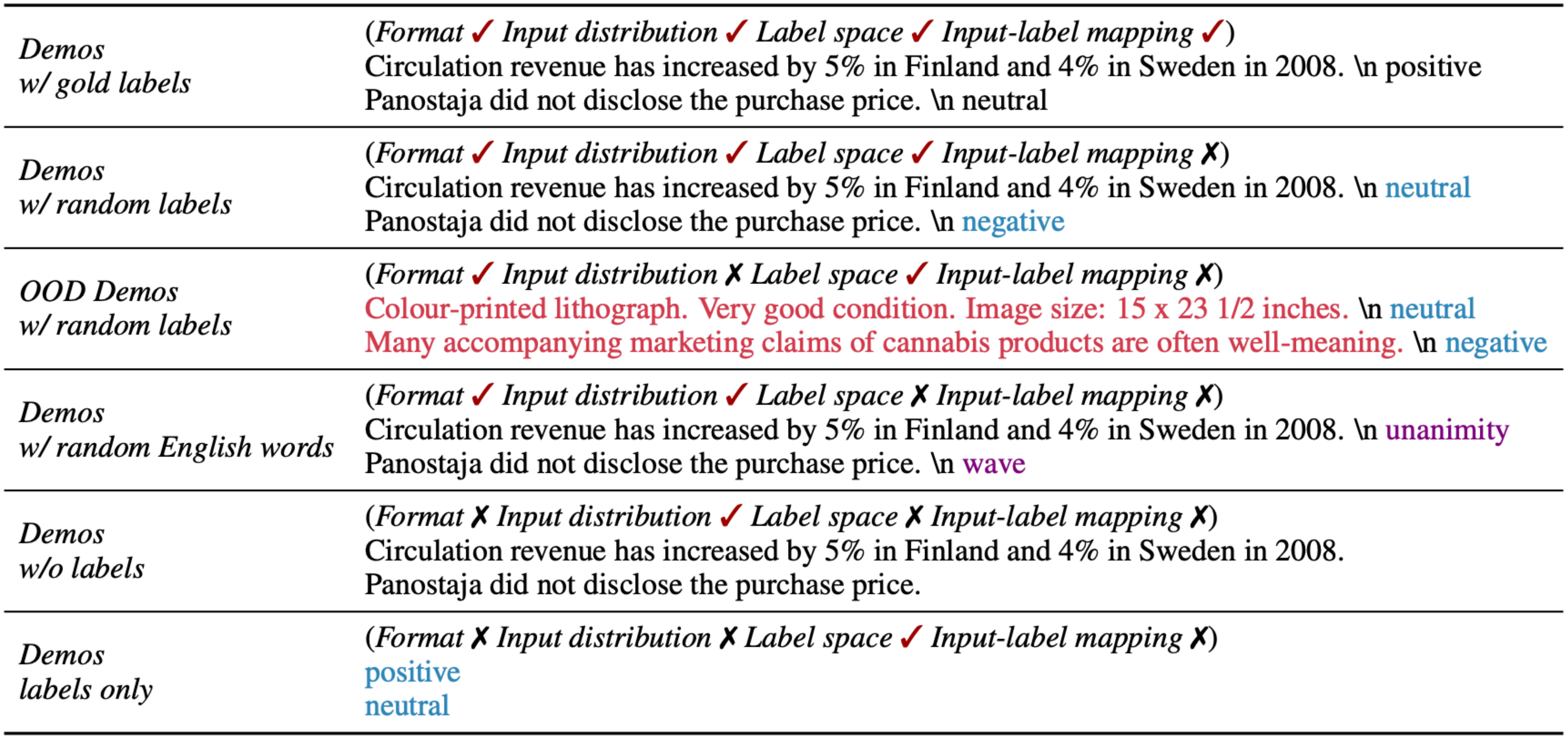
Min et al., EMNLP 2022

How important is input-label mapping?



Min et al., EMNLP 2022

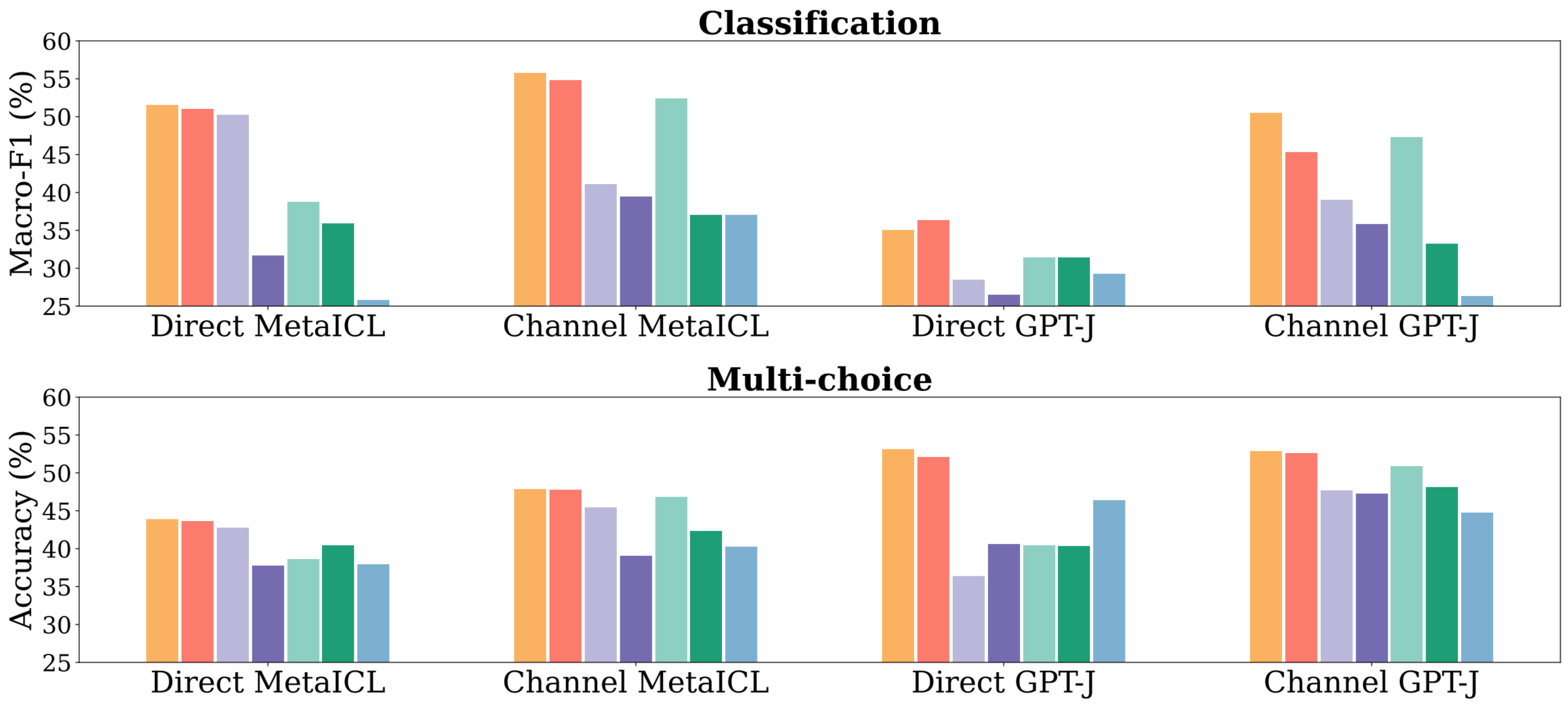
What about other aspects of the demonstrations?



Demos w/ gold labels	(*Format* ✓ *Input distribution* ✓ *Label space* ✓ *Input-label mapping* ✓) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n positive Panostaja did not disclose the purchase price. \n neutral
Demos w/ random labels	(*Format* ✓ *Input distribution* ✓ *Label space* ✓ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n neutral Panostaja did not disclose the purchase price. \n negative
OOD Demos w/ random labels	(*Format* ✓ *Input distribution* ✗ *Label space* ✓ *Input-label mapping* ✗) Colour-printed lithograph. Very good condition. Image size: 15 x 23 1/2 inches. \n neutral Many accompanying marketing claims of cannabis products are often well-meaning. \n negative
Demos w/ random English words	(*Format* ✓ *Input distribution* ✓ *Label space* ✗ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. \n unanimity Panostaja did not disclose the purchase price. \n wave
Demos w/o labels	(*Format* ✗ *Input distribution* ✓ *Label space* ✗ *Input-label mapping* ✗) Circulation revenue has increased by 5% in Finland and 4% in Sweden in 2008. Panostaja did not disclose the purchase price.
Demos labels only	(*Format* ✗ *Input distribution* ✗ *Label space* ✓ *Input-label mapping* ✗) positive neutral

Min et al., EMNLP 2022

What about other aspects of the demonstrations?



	F	*L*	*I*	*M*
🟧 Gold labels	✓	✓	✓	✓
🟥 Random labels	✓	✓	✓	✗
🟪 OOD + Random labels	✓	✓	✗	✗
🟪 Random labels only	✗	✓	✗	✗
🟩 Random English words	✓	✗	✓	✗
🟩 No labels	✗	✗	✓	✗
🟦 No demonstrations	✗	✗	✗	✗

F: Format
L: Label space
I: Input distribution
M: Input-Label Mapping

Min et al., EMNLP 2022

This is a nice study, but it is also limited

- Only studied simple classification tasks
- The effect of these different aspects of demonstrations on *generation* tasks is unclear

Followup work tells a different story:

- Yoo et al., 2022 shows that input-label mappings matter quite significantly when using different experimental conditions and eval metrics
- Madaan and Yazdanbakhsh (2022) show that random rationales degrade chain-of-thought performance, but other modifications to the rationale (e.g., wrong equations) don't affect it too much

Okay, but how does the LLM make use of (any aspect of) the demonstration?

“We investigate the hypothesis that transformer-based in-context learners implement standard learning algorithms implicitly, by encoding smaller models in their activations, and updating these implicit models as new examples appear in the context.”
— Akyürek et al., ICLR 2023

Can a Transformer do linear regression?

- Akyürek et al., ICLR 2023: theoretically proved that LLMs can learn a small linear model

$$\sum_i \mathcal{L}(\boldsymbol{w}^\top \boldsymbol{x}_i, y_i) + \lambda \|\boldsymbol{w}\|_2^2$$

- The above linear regression loss has a *closed form* solution: $\boldsymbol{w}^* = (X^\top X + \lambda I)^{-1} X^\top y$

- But can also be solved via gradient descent

$$\boldsymbol{w}' = \boldsymbol{w} - \alpha \frac{\partial}{\partial \boldsymbol{w}} \Big(\mathcal{L}(\boldsymbol{w}^\top \boldsymbol{x}_i, y_i) + \lambda \|\boldsymbol{w}\|_2^2 \Big) = \boldsymbol{w} - 2\alpha(\boldsymbol{x}\boldsymbol{w}^\top \boldsymbol{x} - y\boldsymbol{x} + \lambda \boldsymbol{w})$$

Transformers can implement some primitive operations useful for learning

It will be useful to first establish a few computational primitives with simple transformer implementations. Consider the following four functions from $\mathbb{R}^{H \times T} \to \mathbb{R}^{H \times T}$:

$\texttt{mov}(H; s, t, i, j, i', j')$: selects the entries of the s^{th} column of H between rows i and j, and copies them into the t^{th} column ($t \geq s$) of H between rows i' and j', yielding the matrix:

$$\begin{bmatrix} | & H_{:i-1,t} & | \\ H_{:,:t} & H_{i':j',s} & H_{:,t+1:} \\ | & H_{j,t} & | \end{bmatrix}.$$

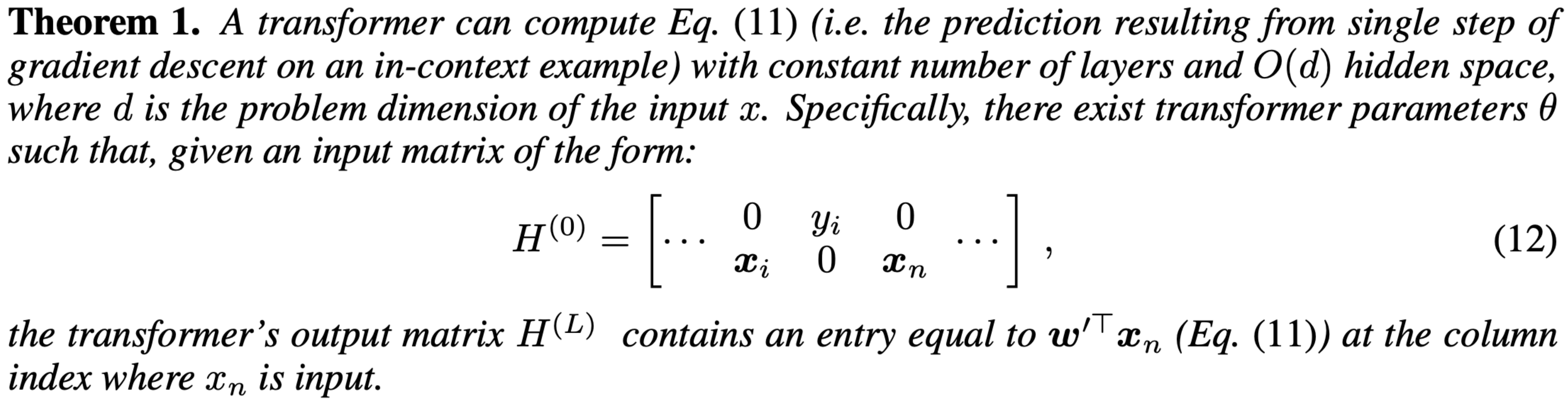
$\texttt{mul}(H; a, b, c, (i, j), (i', j'), (i'', j''))$: in *each* column $\boldsymbol{h}$ of H, interprets the entries between i and j as an $a \times b$ matrix A_1, and the entries between i' and j' as a $b \times c$ matrix A_2, multiplies these matrices together, and stores the result between rows i'' and j'', yielding a matrix in which each column has the form $[\boldsymbol{h}_{:i''-1}, A_1 A_2, \boldsymbol{h}_{j'':}]^\top$.

$\texttt{div}(H; (i, j), i', (i'', j''))$: in each column $\boldsymbol{h}$ of H, divides the entries between i and j by the absolute value of the entry at i', and stores the result between rows i'' and j'', yielding a matrix in which every column has the form $[\boldsymbol{h}_{:i''-1}, \boldsymbol{h}_{i:j}/|\boldsymbol{h}_{i'}|, \boldsymbol{h}_{j'':}]^\top$.

$\texttt{aff}(H; (i, j), (i', j'), (i'', j''), W_1, W_2, b)$: in each column $\boldsymbol{h}$ of H, applies an affine transformation to the entries between i and j and i' and j', then stores the result between rows i'' and j'', yielding a matrix in which every column has the form $[\boldsymbol{h}_{:i''-1}, W_1 \boldsymbol{h}_{i:j} + W_2 \boldsymbol{h}_{i':j'} + b, \boldsymbol{h}_{j'':}]^\top$.

Lemma 1. *Each of* `mov`, `mul`, `div` *and* `aff` *can be implemented by a single transformer decoder layer: in Eq.* (1) *and Eq.* (4)*, there exist matrices* W^Q, W^K, W^V, W^F, W_1 *and* W_2 *such that, given a matrix* H *as input, the layer's output has the form of the corresponding function output above.* [1]

Using these primitives, they show that Transformers can implement one step of gradient descent



Theorem 1. *A transformer can compute Eq.* (11) *(i.e. the prediction resulting from single step of gradient descent on an in-context example) with constant number of layers and* $O(d)$ *hidden space, where* d *is the problem dimension of the input* x*. Specifically, there exist transformer parameters* θ *such that, given an input matrix of the form:*

$$H^{(0)} = \begin{bmatrix} \cdots & 0 & y_i & 0 & \cdots \\ & \boldsymbol{x}_i & 0 & \boldsymbol{x}_n & \end{bmatrix}, \tag{12}$$

the transformer's output matrix $H^{(L)}$ *contains an entry equal to* $\mathbf{w}'^\top \boldsymbol{x}_n$ *(Eq.* (11)*) at the column index where* x_n *is input.*

Probe networks show that Transformers encode useful intermediate computations while training the linear models

To do so, we identify two intermediate quantities that we expect to be computed by gradient descent and ridge-regression variants: the **moment vector** $X^\top Y$ and the (min-norm) least-square estimated **weight vector** $\boldsymbol{w}_{\text{OLS}}$, each calculated after feeding n exemplars. We take a trained in-context learner, freeze its weights, then train an auxiliary **probing** model (Alain & Bengio, 2016) to attempt to recover the target quantities from the learner's hidden representations. Specifically, the probe model takes hidden states at a layer $H^{(l)}$ as input, then outputs the prediction for target variable. We define a probe with **position-attention** that computes (Appendix E):

$$\boldsymbol{\alpha} = \text{softmax}(\boldsymbol{s}_v) \tag{20}$$

$$\hat{\boldsymbol{v}} = \text{FF}_{\text{v}}(\boldsymbol{\alpha}^\top W_v H^{(l)}) \tag{21}$$