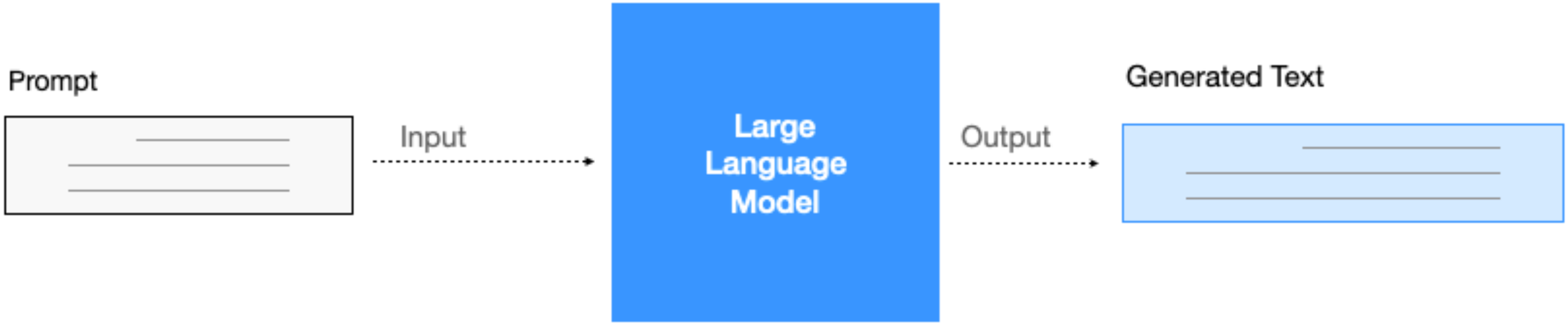
DETAIL: TASK DEMONSTRATION ATTRIBUTION FOR INTERPRETABLE IN-CONTEXT LEARNING

Presenter: Zijian ZHOU

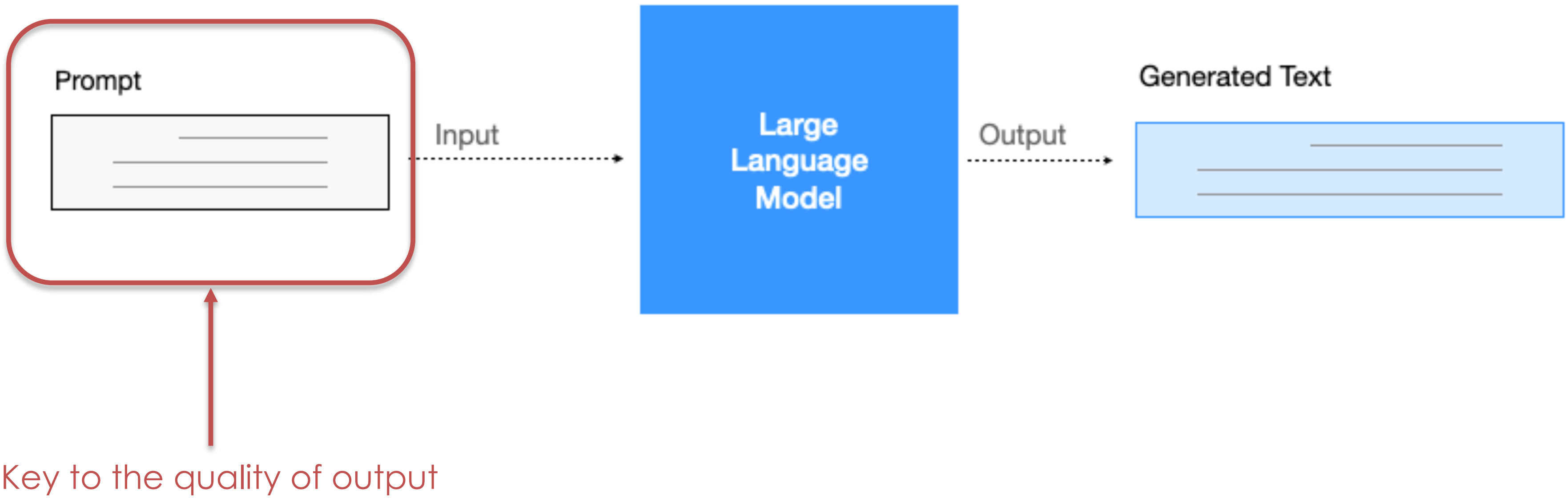
NeurIPS 2024

Large Language Models



Use prompts to help improve model's response to user query

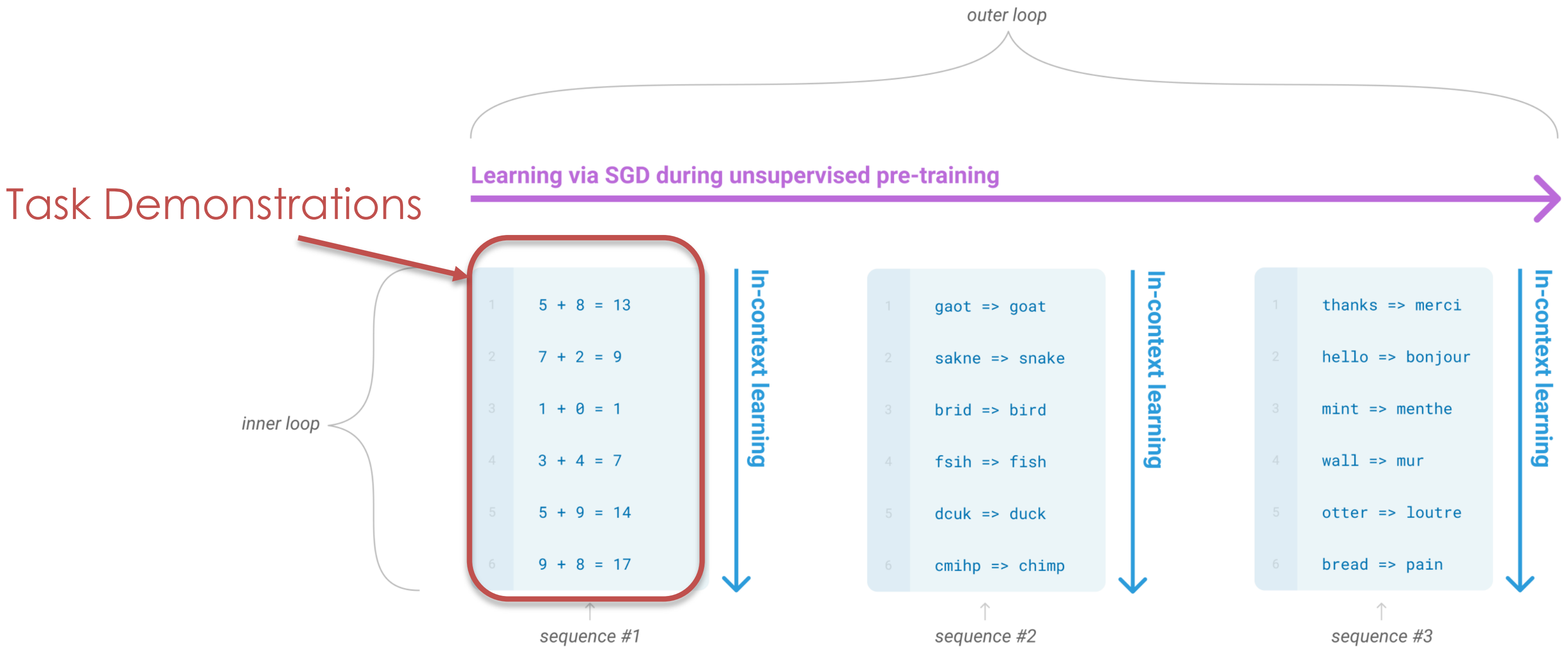
Large Language Models



In-context Learning (ICL)

SMART
Singapore-MIT Alliance for Research and Technology

NUS National University of Singapore



Source: Brown et. al., Language Model are Few-shot Learners, in NeuRIPS 2020.

LLM learns to perform arithmetic and manipulate spelling
During **test-time** using **ICL**.

In-context Learning (ICL)

An analogy between classic ML and ICL.

Classic ML: Requires training data points to train ML.

ICL: Task demonstrations supplied to during inference.

Attributing ICL

In ICL, we are interested in

- Save cost => Demonstration Selection
- Speed => Computable at inference time
- Interpretable => Can we interpret the attribution score?

Attributing ICL

Can we use existing attribution methods for ICL?

- Quality => Good Demonstration Selection
- Speed => Computable at inference time
- Interpretable => Can we interpret the attribution score?

DETAIL

Using DETAIL for ICL

- Quality => Good Demonstration Selection
- Speed => Computable at inference time
- Interpretable => Can we interpret the attribution score?

Preliminaries

Influence Function [1]: Computes the "influence" of a training data point on the prediction of a test data point.

$$\mathcal{I}(z_i, z_{\text{test}}) \coloneqq \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top \mathcal{I}_{\text{reg}}(z_i) = \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z_i, \hat{\theta})$$

[1] Koh et. al., Understanding Black-box Predictions via Influence Functions, ICML 2017

Preliminaries

Influence Function [1]: Computes the "influence" of a training data point on the prediction of a test data point.

Test data point

Gradient on test data

Gradient on train data

$$\boxed{\mathcal{I}}(\boxed{z_i},\boxed{z_{\text{test}}}) := \nabla_\theta L(z_{\text{test}},\hat{\theta})^\top \mathcal{I}_{\text{reg}}(z_i) = \boxed{\nabla_\theta L(z_{\text{test}},\hat{\theta})^\top}\boxed{H_{\hat{\theta}}^{-1}}\boxed{\nabla_\theta L(z_i,\hat{\theta})}$$

Score

Train data point

Inverse hessian of model parameters

[1] Koh et. al., Understanding Black-box Predictions via Influence Functions, ICML 2017

Preliminaries

Can we apply the influence function to attribute ICL demonstrations?

Preliminaries

Can we apply the influence function to attribute ICL demonstrations?

Not directly. There is no "gradient" for ICL.

$$\mathcal{I}(z_i, z_{\text{test}}) := \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top \mathcal{I}_{\text{reg}}(z_i) = \boxed{\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z_i, \hat{\theta})}$$

Preliminaries

Reflection: How does ICL achieves “learning”?

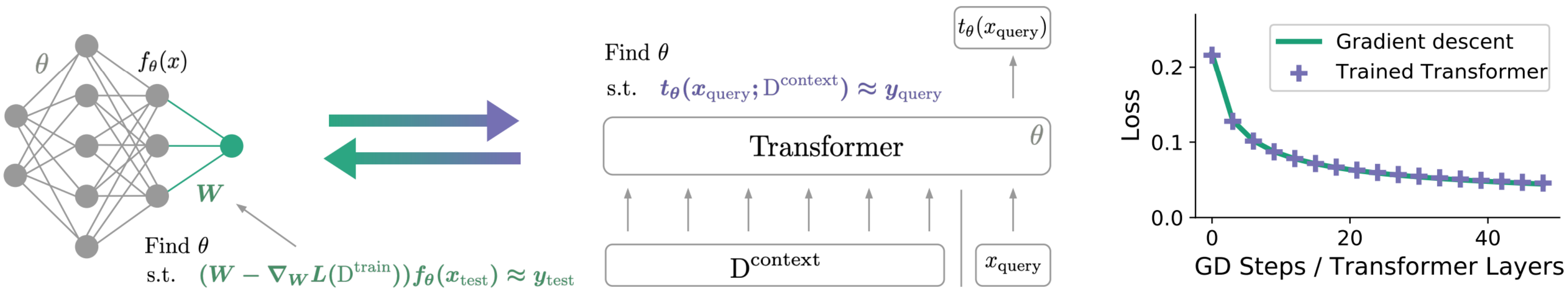
Preliminaries

Reflection: How does ICL achieves “learning”?

- While there is no explicit parameter update,
- There is indeed an implicit gradient descent.

Preliminaries

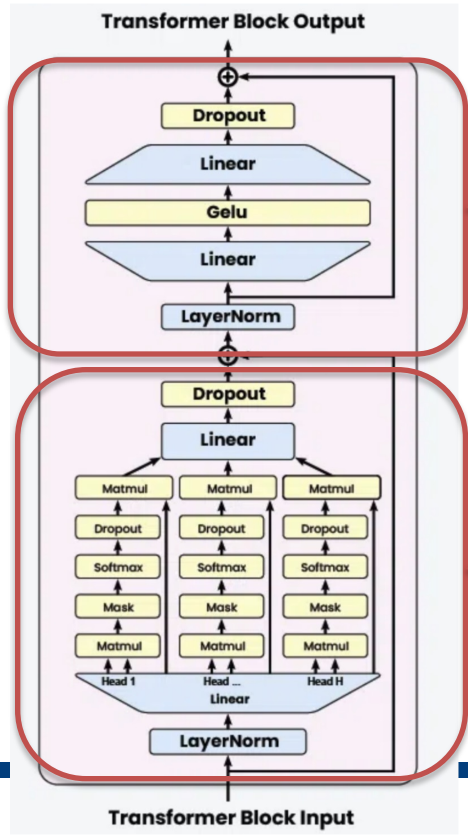
Equivalence between gradient descent and ICL



Source: Oswald et. al., Transformers Learn In-Context by Gradient Descent, ICML 2023

Preliminaries

Transformers can also learn non-linear regression tasks using a deep representation of data



Create deep representation

"Gradient descent"

Preliminaries

Specific formulation [1]:

Proposition 2. *Given a Transformer block i.e. a MLP* $m(e)$ *which transforms the tokens* $e_j = (x_j, y_j)$ *followed by an attention layer, we can construct weights that lead to gradient descent dynamics descending* $\frac{1}{2N}\sum_{i=1}^{N} ||Wm(x_i) - y_i||^2$. *Iteratively applying Transformer blocks therefore can solve kernelized least-squares regression problems with kernel function* $k(x, y) = m(x)^\top m(y)$ *induced by the MLP* $m(\cdot)$.

[1] Source: Oswald et. al., Transformers Learn In-Context by Gradient Descent, ICML 2023

DETAIL

A (L2 regularized) kernelized regression:

$$L(x, y) = [m(x)\beta - y]^2 + \lambda\beta^\top\beta \, .$$

$m(x)$: the hidden state of a transformer layer

β : weight factor of the kernelized feature

λ : a regularization term

DETAIL

A (L2 regularized) kernelized regression:

$$\boxed{L(x, y)} = [m(x)\beta - y]^2 + \lambda\beta^\top\beta\,.$$

DETAIL

Singapore-MIT Alliance for Research and Technology

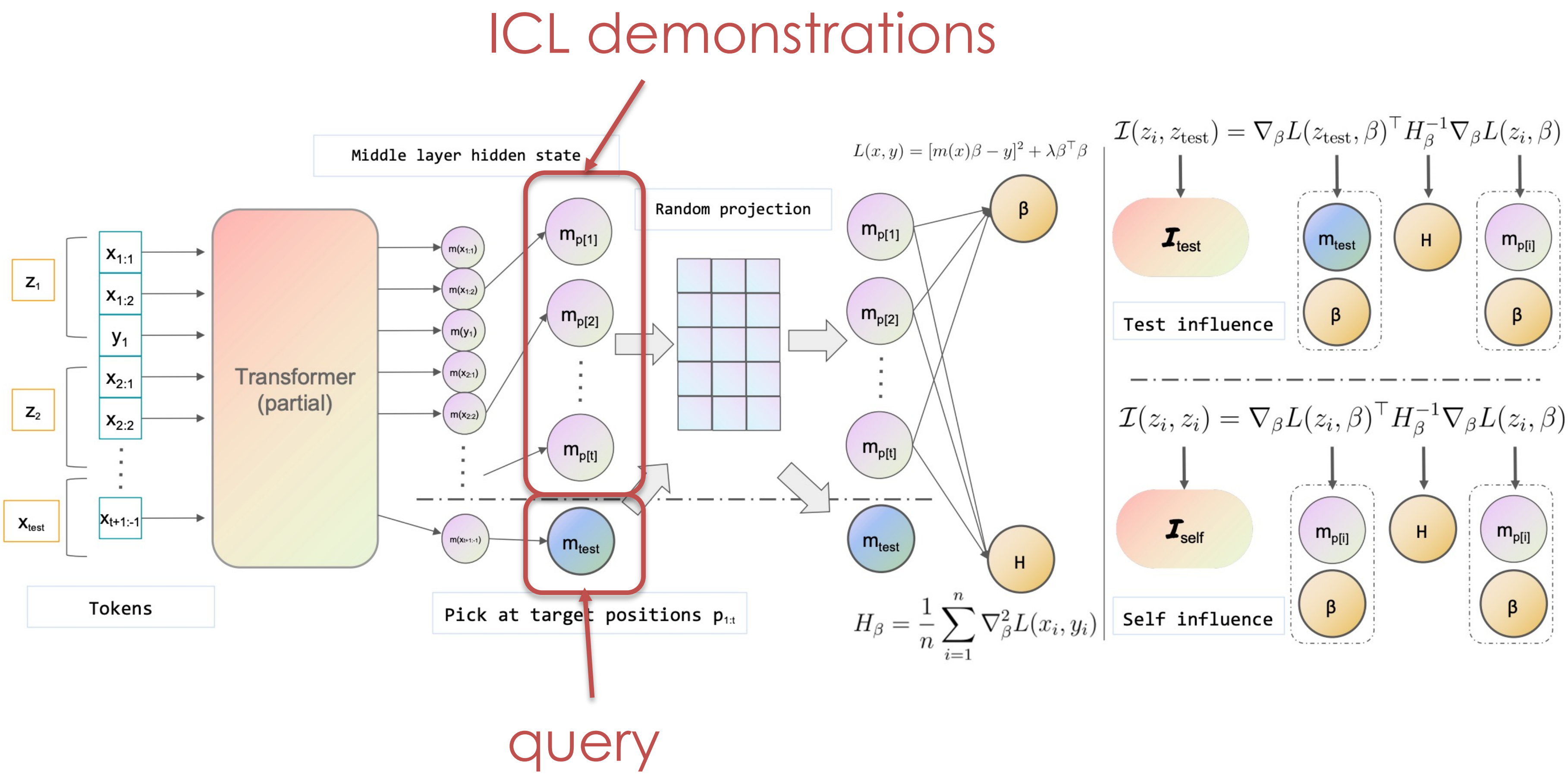
National University of Singapore

Reformulate the influence function

$$\boxed{L(x,y)} = [m(x)\beta - y]^2 + \lambda\beta^\top\beta\,.$$

$$\begin{aligned}\mathcal{I}(z_{\text{test}}, z) &:= \boxed{\nabla_\beta L(x_{\text{test}}, y_{\text{test}})^\top} \mathcal{I}_{\text{reg}}(z) \\ &= n[m(x_{\text{test}})^\top(m(x_{\text{test}})\beta - y_{\text{test}}) + \lambda\beta](K + \lambda I)^{-1}[m(x)^\top(m(x)\beta - y) + \lambda\beta]\end{aligned}$$

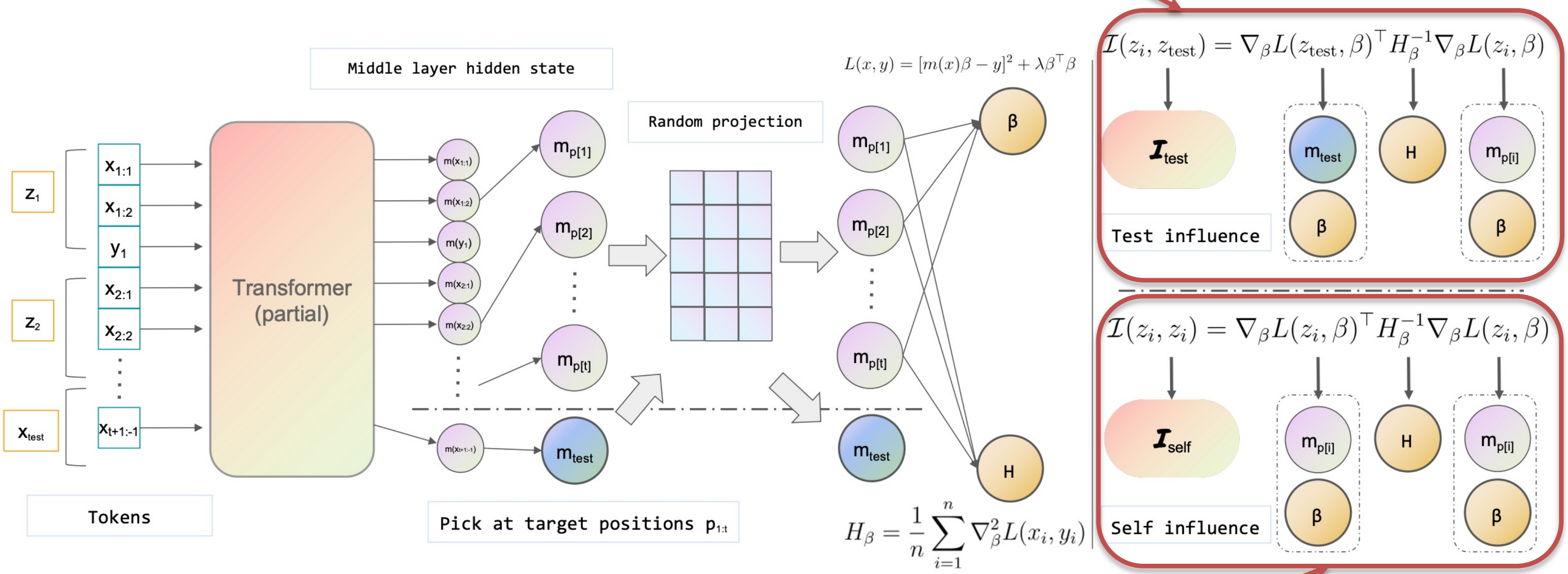
DETAIL



ICL demonstrations
Middle layer hidden state
Random projection
$L(x,y) = [m(x)\beta - y]^2 + \lambda\beta^\top\beta$
$\mathcal{I}(z_i, z_{\text{test}}) = \nabla_\beta L(z_{\text{test}}, \beta)^\top H_\beta^{-1} \nabla_\beta L(z_i, \beta)$
z_1
z_2
x_{test}
Transformer (partial)
Tokens
Pick at target positions $p_{1:t}$
$H_\beta = \frac{1}{n}\sum_{i=1}^{n} \nabla_\beta^2 L(x_i, y_i)$
Test influence
$\mathcal{I}(z_i, z_i) = \nabla_\beta L(z_i, \beta)^\top H_\beta^{-1} \nabla_\beta L(z_i, \beta)$
Self influence
query

DETAIL

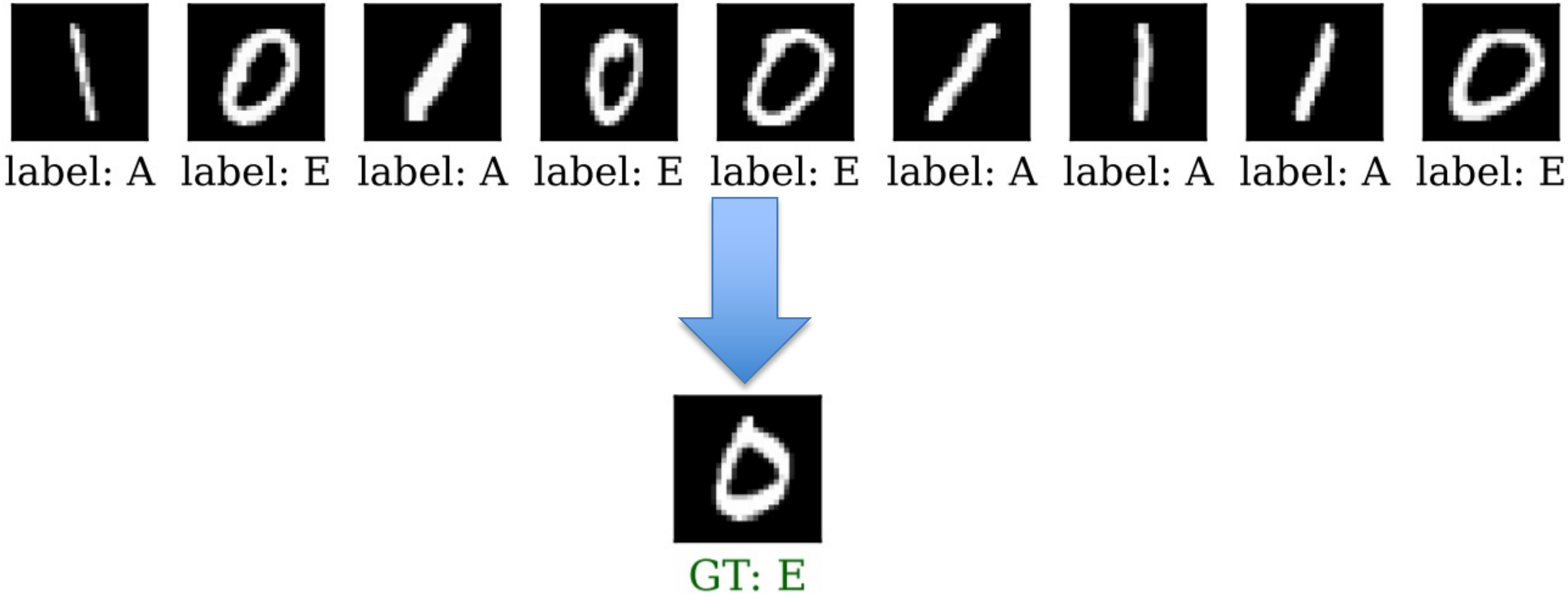
Score against query



Score against self

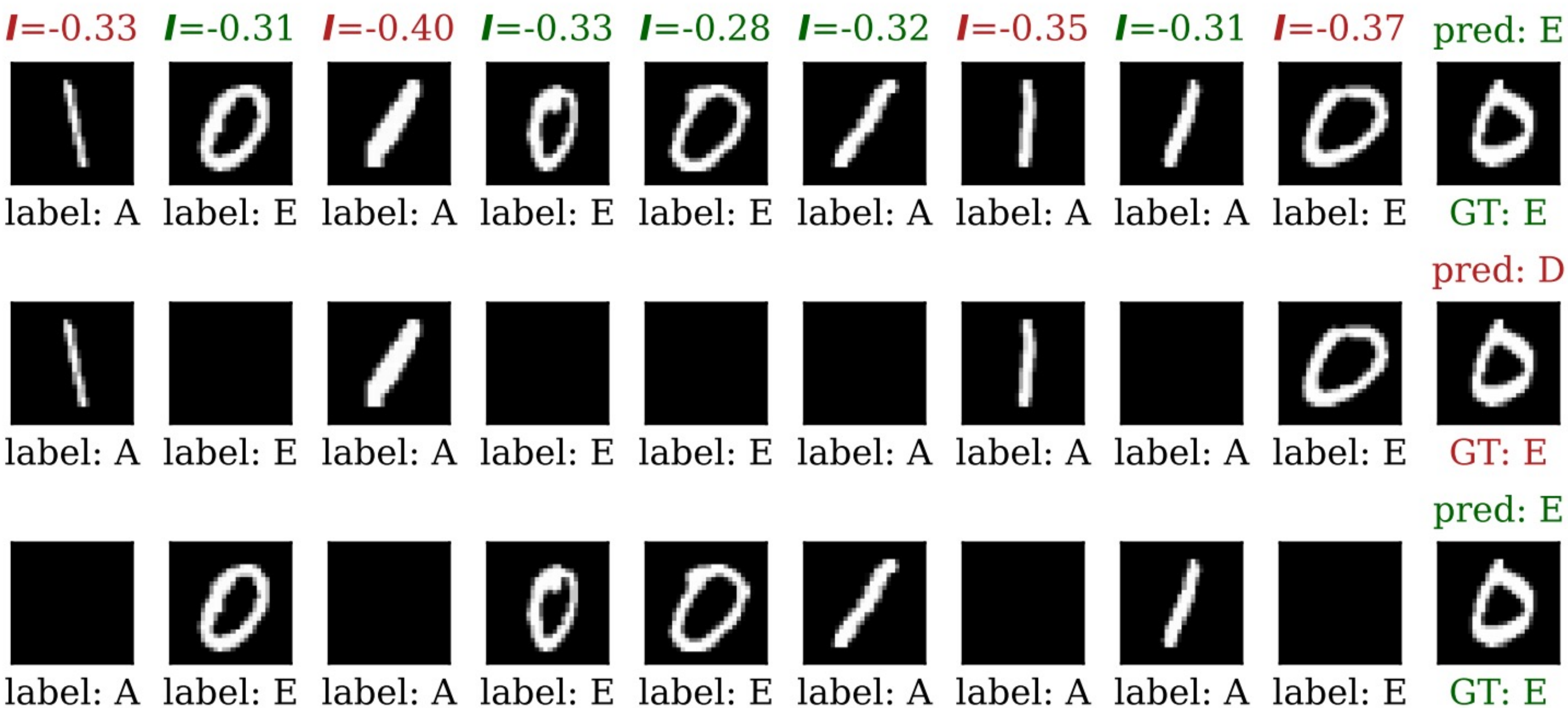
Empirical investigation

Custom transformer: predicting the label of MNIST digits using ICL



Empirical investigation

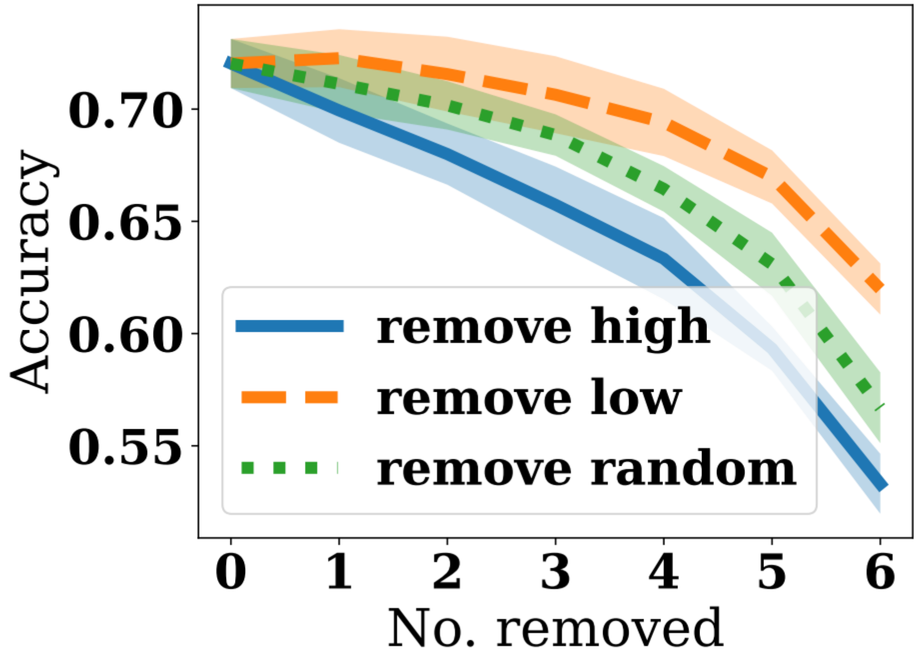
Remove some of the ICL demonstrations according to DETAIL (test influence) score.



Experiments

Empirical investigation

Remove some of the ICL demonstrations according to DETAIL (test influence) score.



Experiments

Moving on to LLM

We consider two ICL-related tasks.

- Noisy label detection
- Demonstration order optimization
- Demonstration curation

Noisy Label Detection

ICL demonstrations may contain corrupted samples.

e.g.

1 + 3 = 4;

\## 2 + 5 = 7;

\## 4 + 2 = 8; Corrupted sample!

\## 10 – 3 = 7;

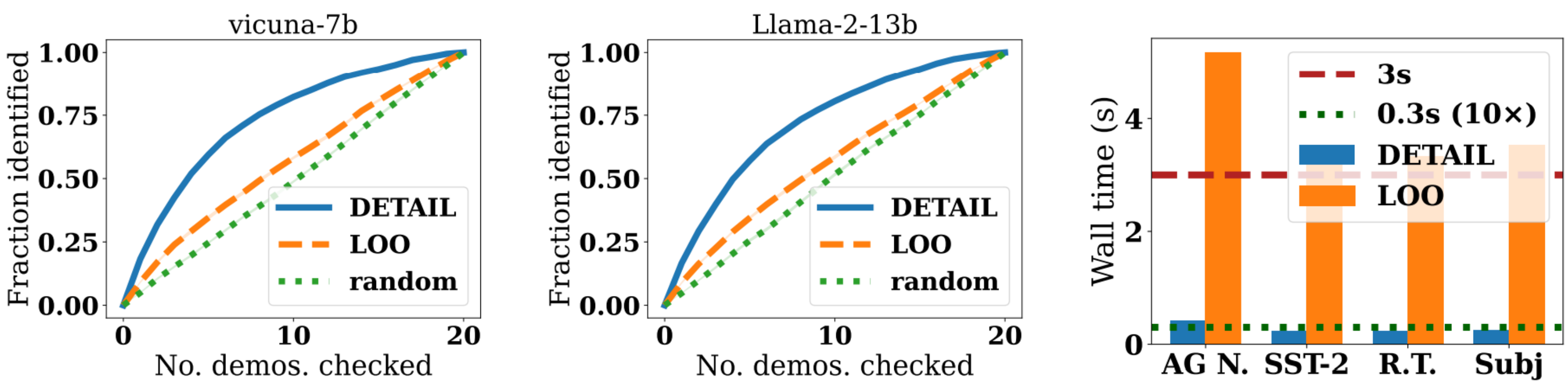
\## 42 + 2 = 44;

\## 23 + 2 = [__]

Noisy Label Detection

Use DETAIL (self influence) to detect corrupted samples. High DETAIL (self) score => bad sample.

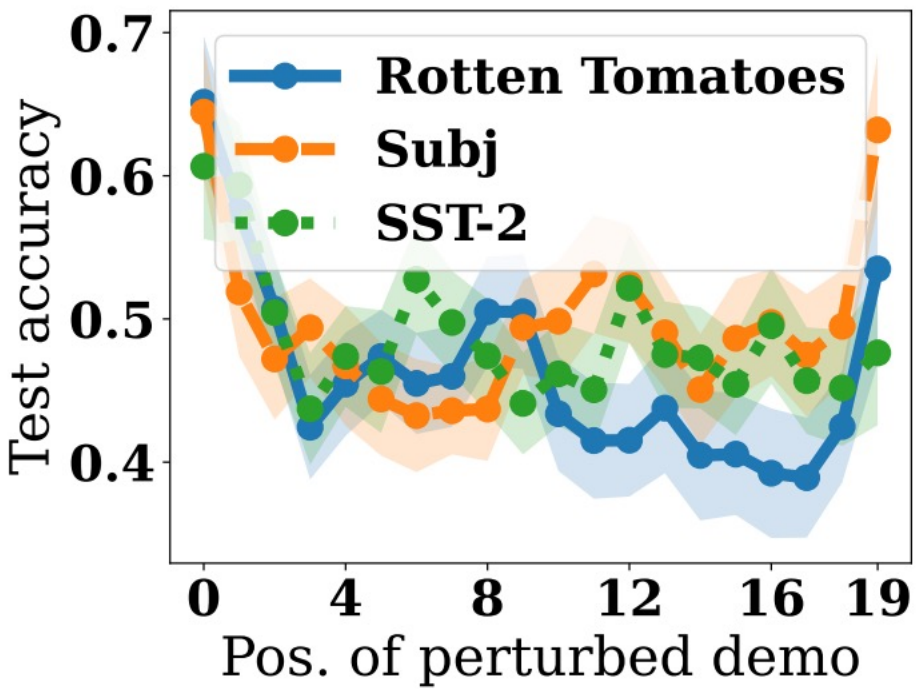
Much better (both quality and speed) than Leave-one-out (LOO).



Experiments

Order Optimization

Different orders of demonstrations can lead to varied performance.

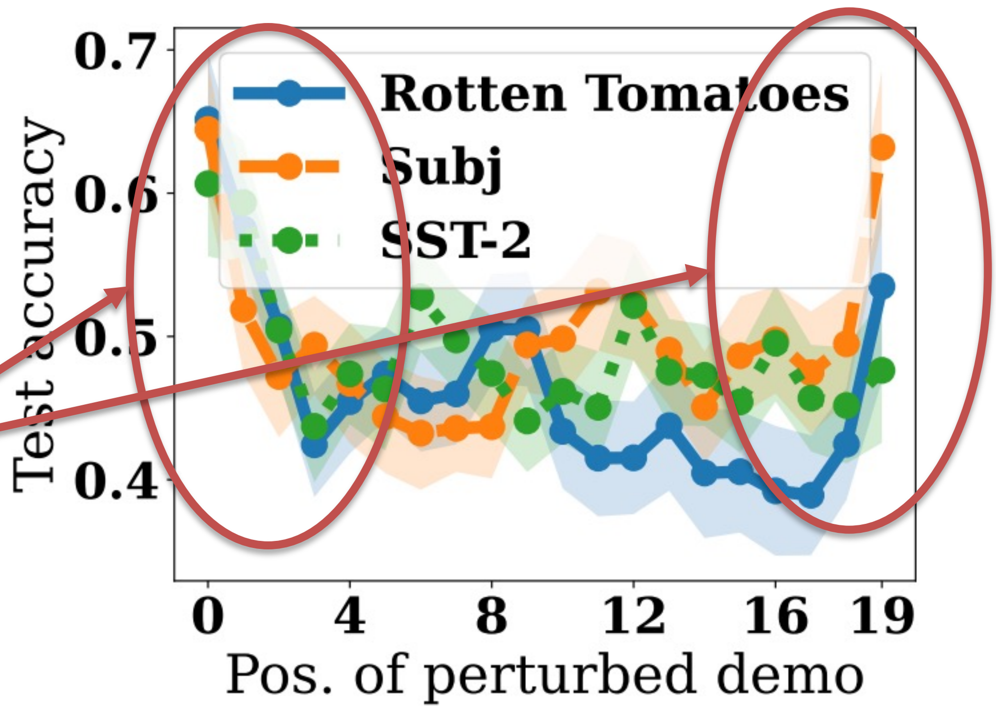


Experiments

Order Optimization

We should place demonstrations with bad quality at the two ends

Higher acc at two ends



Experiments

Order Optimization

We should place demonstrations with high DETAIL scores (self influence) at the two ends

	Subj	SST-2	Rotten Tomatoes
No corrupted demo			
Baseline (random)	0.722 (7.22e-03)	0.665 (5.24e-03)	0.660 (1.08e-02)
Reorder (`DETAIL`)	0.743 (7.10e-03)	0.679 (5.42e-03)	0.684 (1.15e-02)
Difference ↑	**0.0206 (7.40e-03)**	**0.0139 (6.08e-03)**	**0.0244 (1.11e-02)**
Corrupt 3 demos			
Baseline (random)	0.655 (8.54e-03)	0.607 (7.61e-03)	0.553 (1.10e-02)
Reorder (`DETAIL`)	0.685 (9.39e-03)	0.630 (7.04e-03)	0.582 (1.42e-02)
Difference ↑	**0.0300 (9.10e-03)**	**0.0230 (7.22e-03)**	**0.0291 (1.06e-02)**

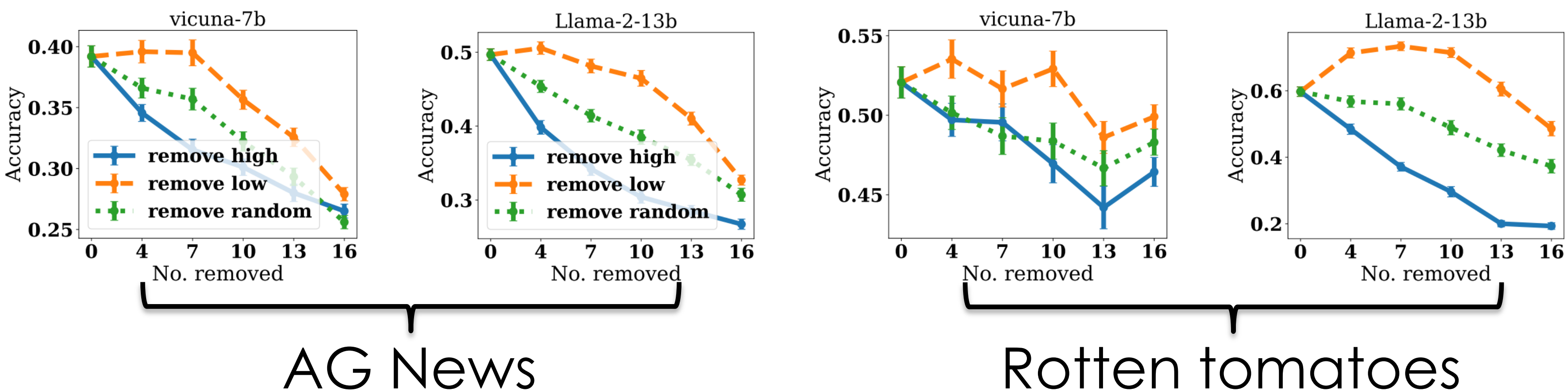
Demonstration Curation

SMART
Singapore-MIT Alliance for Research and Technology

NUS National University of Singapore

We can also use DETAIL scores (test influence) to curate effective demonstrations against a test query set.

Curating using DETAIL shows consistent improvement on various models.



Comparison with other methods

Our method is superior both in terms of speed and attribution quality.

[38, 42] are LOO-based methods

Metric	DETAIL ($d' = 1000$)	IG [45]	LIME [41]	[38]	[42]	Datamodel [13]	Random
Subj							
Accuracy ↑	**0.747 (2.60e-02)**	0.658 (2.22e-02)	0.665 (2.41e-02)	0.583 (2.75e-02)	0.556 (1.38e-02)	0.658 (2.62e-02)	0.654 (2.54e-02)
Wall time ↓	**5.22 (1.17e-01)**	593 (1.20e+01)	393 (2.44e+01)	54.3 (3.78e-01)	9.37 (4.19e-01)	746 (3.42e+00)	N.A.
SST-2							
Accuracy ↑	**0.607 (2.12e-02)**	0.458 (2.06e-02)	0.476 (1.87e-02)	0.513 (1.88e-02)	0.493 (1.34e-02)	0.460 (2.36e-02)	0.469 (2.15e-02)
Wall time ↓	**4.88 (1.35e-01)**	458 (7.99e+00)	337 (1.69e+01)	121 (4.79e+00)	10.6 (7.80e-01)	713 (1.96e+00)	N.A.
Rotten Tomatoes							
Accuracy ↑	**0.555 (1.94e-02)**	0.442 (2.13e-02)	0.435 (1.39e-02)	0.520 (2.17e-02)	0.498 (1.72e-02)	0.484 (1.87e-02)	0.457 (2.19e-02)
Wall time ↓	**5.11 (1.06e-01)**	525 (1.23e+01)	245 (6.32e+01)	122 (4.68e+00)	9.74 (5.57e-01)	732 (2.10e+00)	N.A.
AG News							
Accuracy ↑	**0.412 (1.35e-02)**	0.351 (1.65e-02)	0.368 (1.73e-02)	0.392 (1.42e-02)	0.361 (1.83e-02)	0.373 (1.31e-02)	0.379 (1.70e-02)
Wall time ↓	10.4 (1.07e-01)	1208 (2.16e+01)	599 (1.03e+01)	81.3 (6.05e-01)	**6.94 (4.78e-02)**	997 (7.55e+00)	N.A.

Transferability

Many LLMs we use are black-box

DETAIL requires access to the model internal mechanism

Transferability

DETAIL scores obtained on white-box models can be TRANSFERRED to black-box models!

$\mathcal{I}$

Vicuna-7b DETAIL score GPT-3.5

Experiments

Transferability

DETAIL scores obtained on white-box models can be TRANSFERRED to black-box models!

Dataset	DETAIL ($d' = 1000$)	Random
Subj	**0.842 (2.16e-02)**	0.660 (3.47e-02)
SST-2	**0.812 (1.96e-02)**	0.618 (5.51e-02)
Rotten Tomatoes	**0.690 (4.66e-02)**	0.420 (5.14e-02)
AG News	**0.515 (3.08e-02)**	0.447 (2.73e-02)

Discussions

- DETAIL is a fast, accurate, and interpretable attribution technique designed for transformers.

Discussions

- DETAIL is a fast, accurate, and interpretable attribution technique designed for transformers.
- One limitation is to need to access the model internal mechanism, although empirically, the scores are transferable.

Discussions

- DETAIL is a fast, accurate, and interpretable attribution technique designed for transformers.

- One limitation is to need to access the model internal mechanism, although empirically, the scores are transferable.

- Can it work for more generalized prompting (e.g. CoT)?

Thank you :)