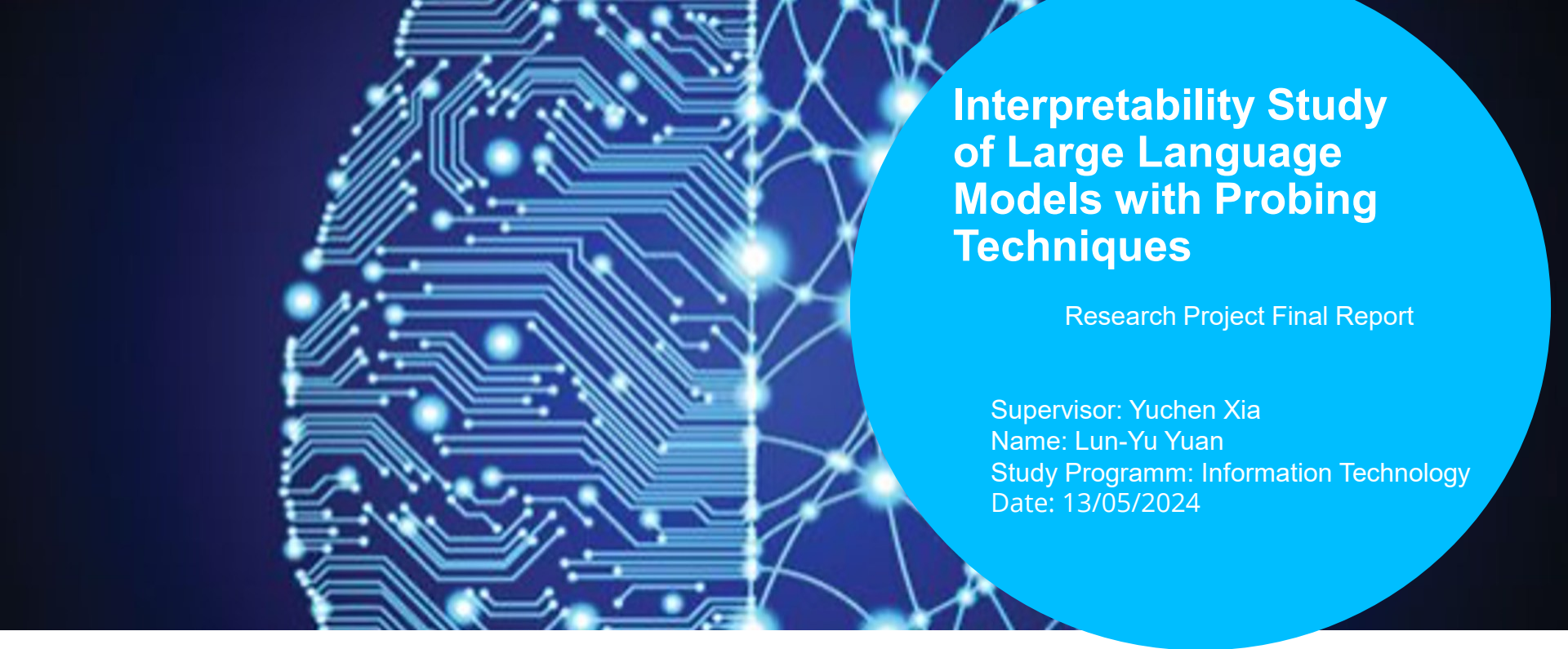
University of Stuttgart
Institute of Industrial Automation
and Software Engineering



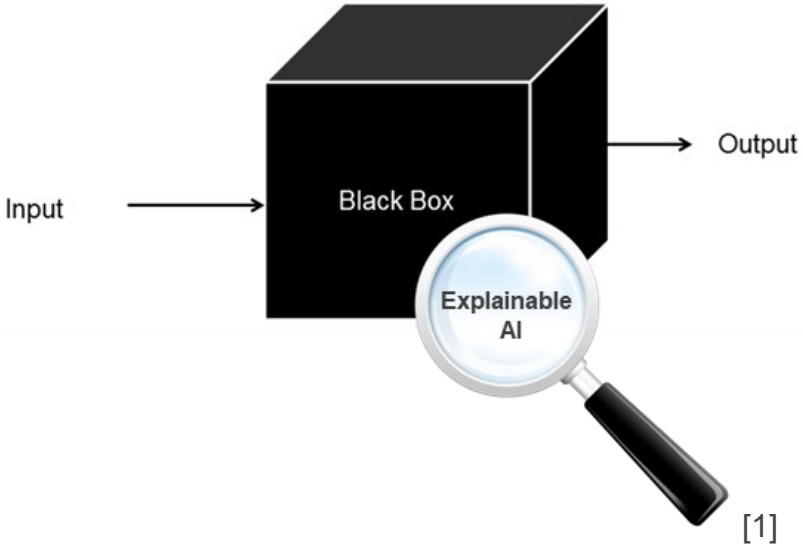
Interpretability Study of Large Language Models with Probing Techniques

Research Project Final Report

Supervisor: Yuchen Xia
Name: Lun-Yu Yuan
Study Programm: Information Technology
Date: 13/05/2024

Motivation & Basis

Interpret the Black Box of Large Language Models (LLMs)



[1]

- **Black box**
  - What is happening during the text generation? (***Transparency***)
  - Is its output what we really want? (***fairness*** and ***beneficial outcomes***)
- **Model Understanding and Improvement**
  - Understand the inner mechanisms of Model
  - Improvement of model development

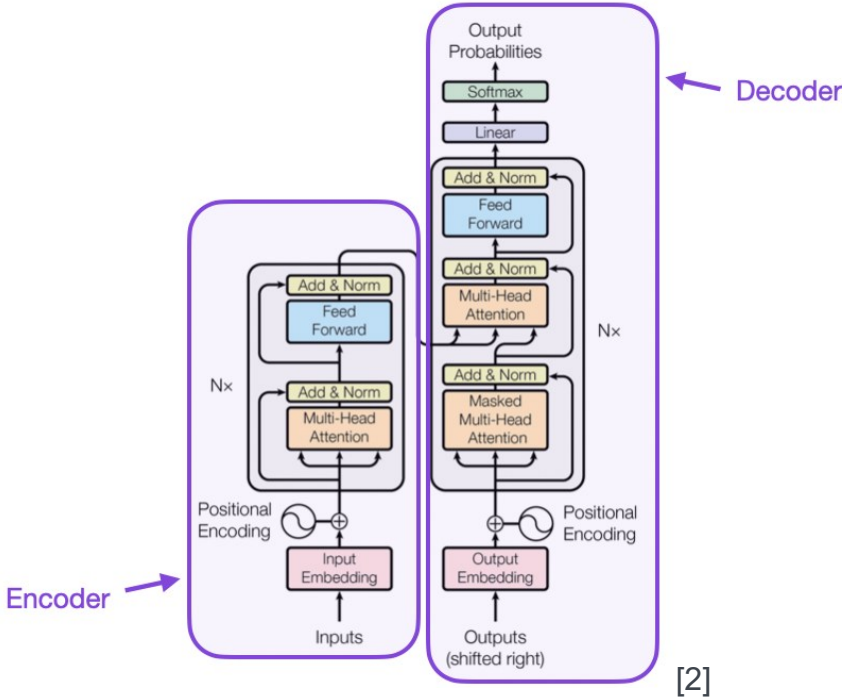
Agenda

- Motivation & Basis
- Conceptual Design
- Implementation
- Evaluation
- Summary and Outlook

Motivation & Basis

Motivation & Basis

Large Language Model Transformer Architecture

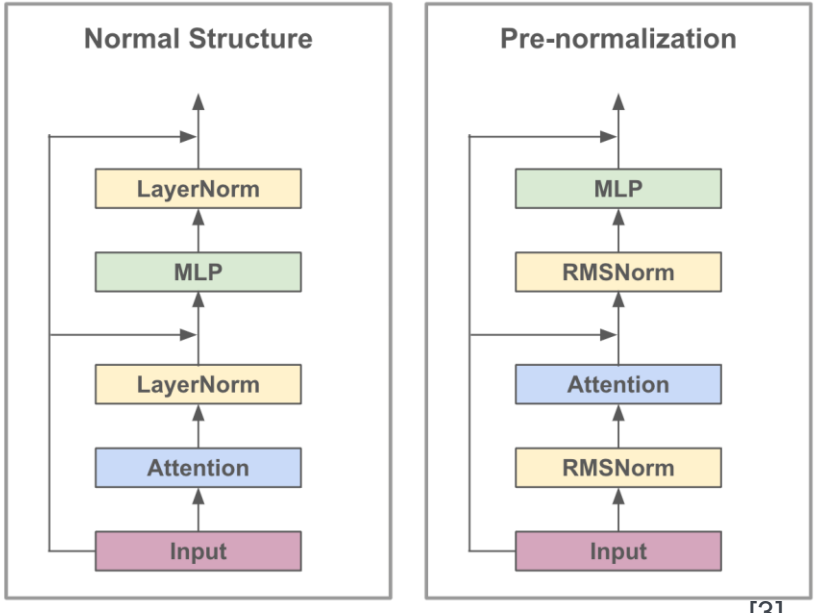


[2]

- The Transformer Model architecture was introduced in the paper “Attention is all you need,” which consists of “Encoder” and “Decoder.”
- Encoder-only LLM example: BERT (embedding)
- Decoder-only LLM example: GPT4, LlaMA-2. (text generation)

Motivation & Basis

Post-Normalization and Pre-Normalization Architecture

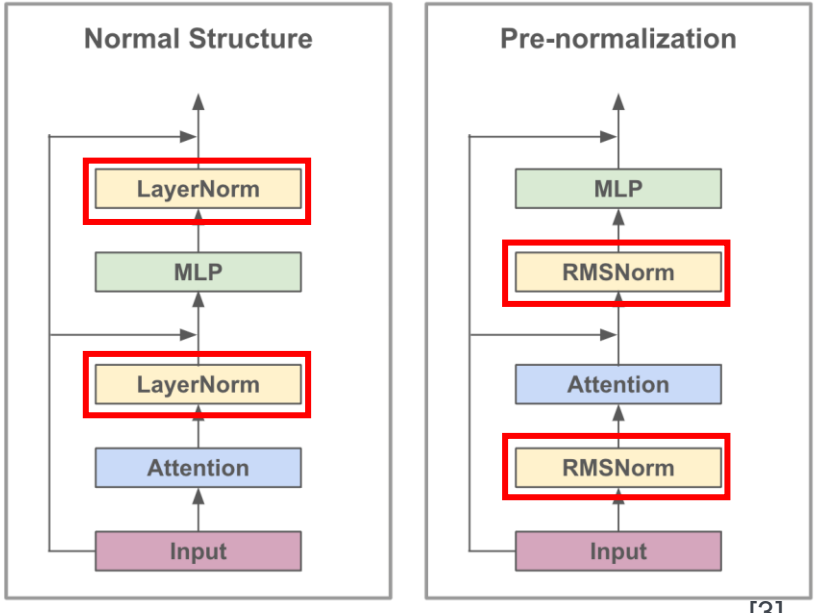


[3]

*RMS (Root Mean Square) Normalization is a simplification of the original LayerNorm.

Motivation & Basis

Post-Normalization and Pre-Normalization Architecture

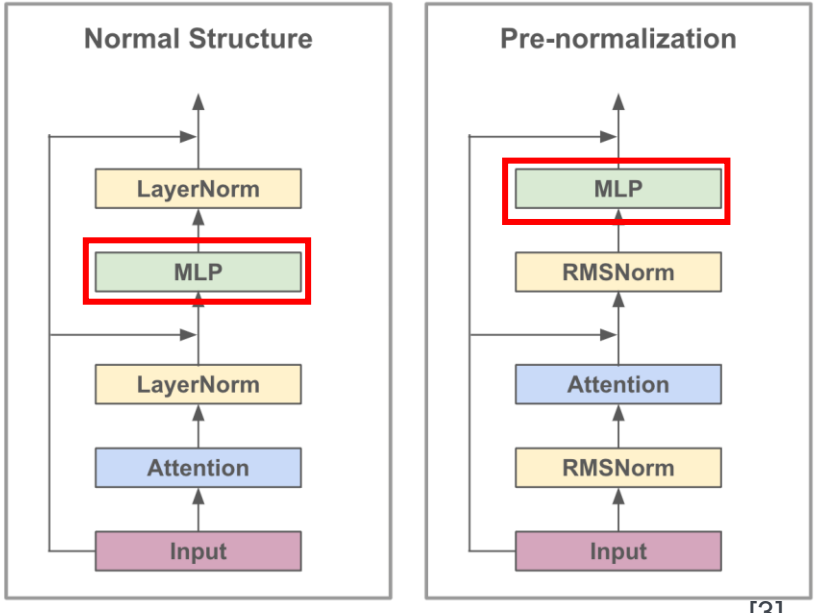


[3]

*RMS (Root Mean Square) Normalization is a simplification of the original LayerNorm.

Motivation & Basis

Post-Normalization and Pre-Normalization Architecture

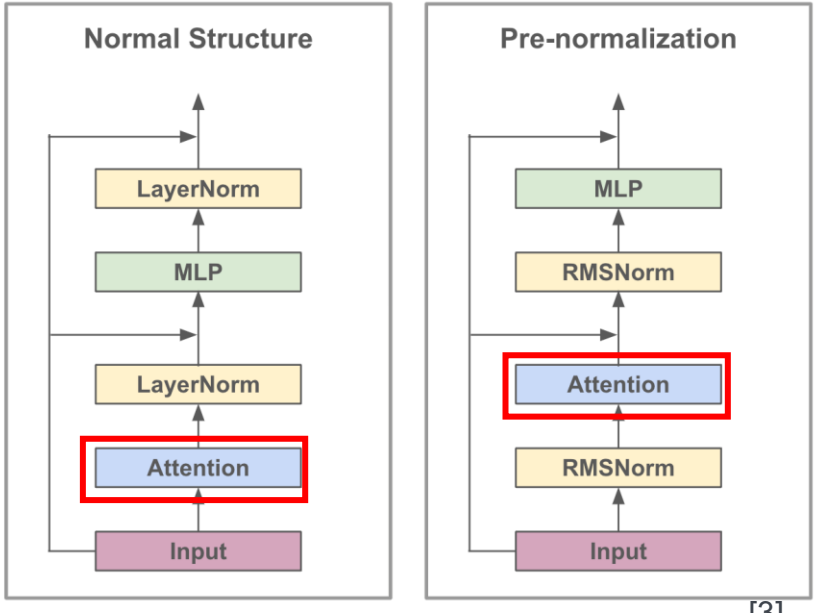


[3]

*RMS (Root Mean Square) Normalization is a simplification of the original LayerNorm.

Motivation & Basis

Post-Normalization and Pre-Normalization Architecture

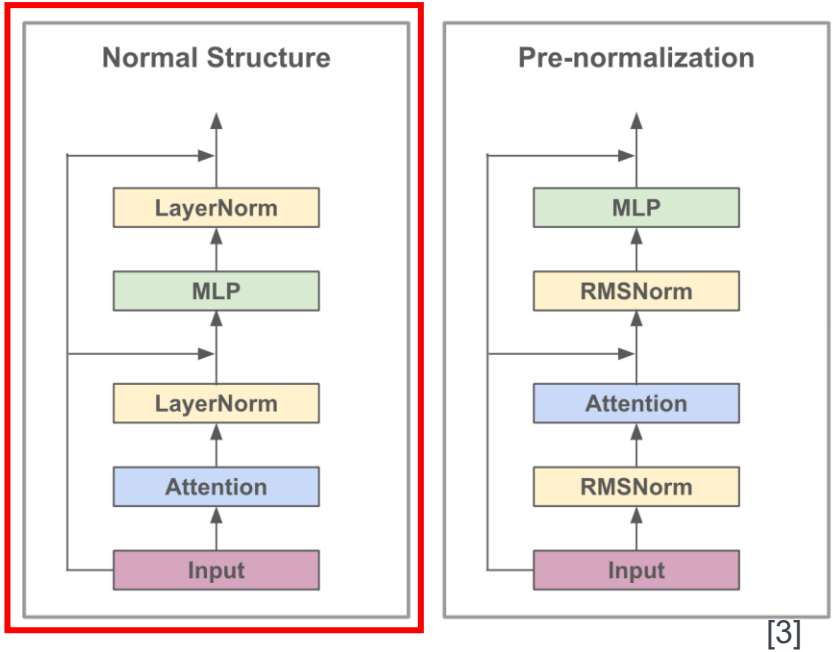


[3]

*RMS (Root Mean Square) Normalization is a simplification of the original LayerNorm.

Motivation & Basis

Post-Normalization and Pre-Normalization Architecture



[3]

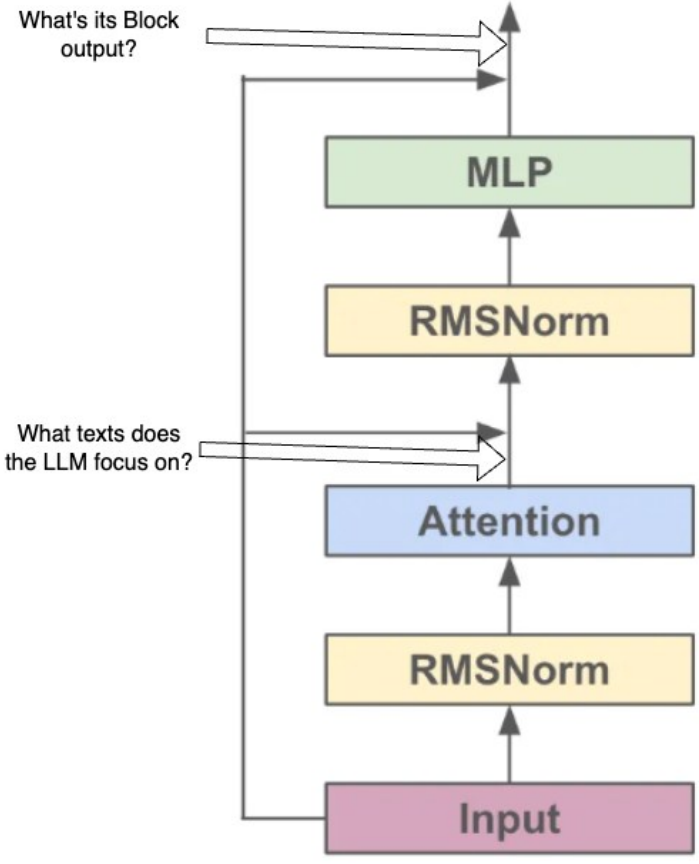
	Normal(Post-normalization)	Pre-normalization
Norm position	LayerNorms **after** Attention and MLP	*RMSNorms (LayerNorms) **before** Attention and MLP
Focus	Emphasizes normalization after each major operation to stabilize the learning process.	Potentially improving training stability and speed.

- LlaMA-2 uses a pre-normalization architecture.
- DeepNorm combines the good performance of Post-LN and the training stability of Pre-LN.

*RMS (Root Mean Square) Normalization is a simplification of the original LayerNorm.

Motivation & Basis

Understand Large Language Model Thoughts



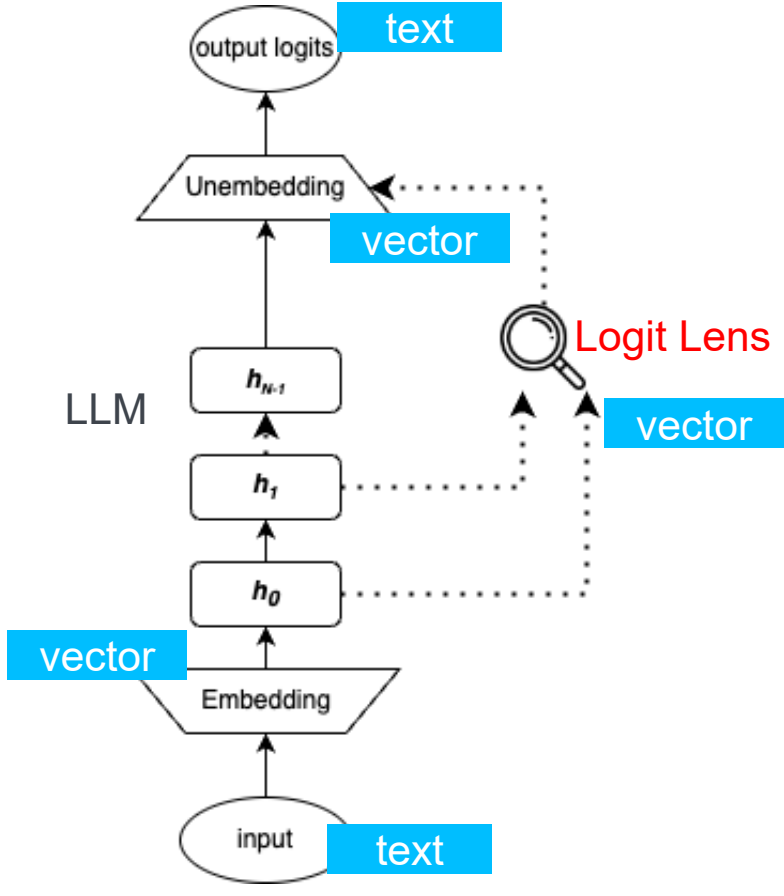
[3]

- Interpreting outputs of each hidden layer's attention mechanism output and block output.
- Attention mechanism output – what texts does the LLM focus on?
- Block output – what results does the LLM generate?

Motivation & Basis

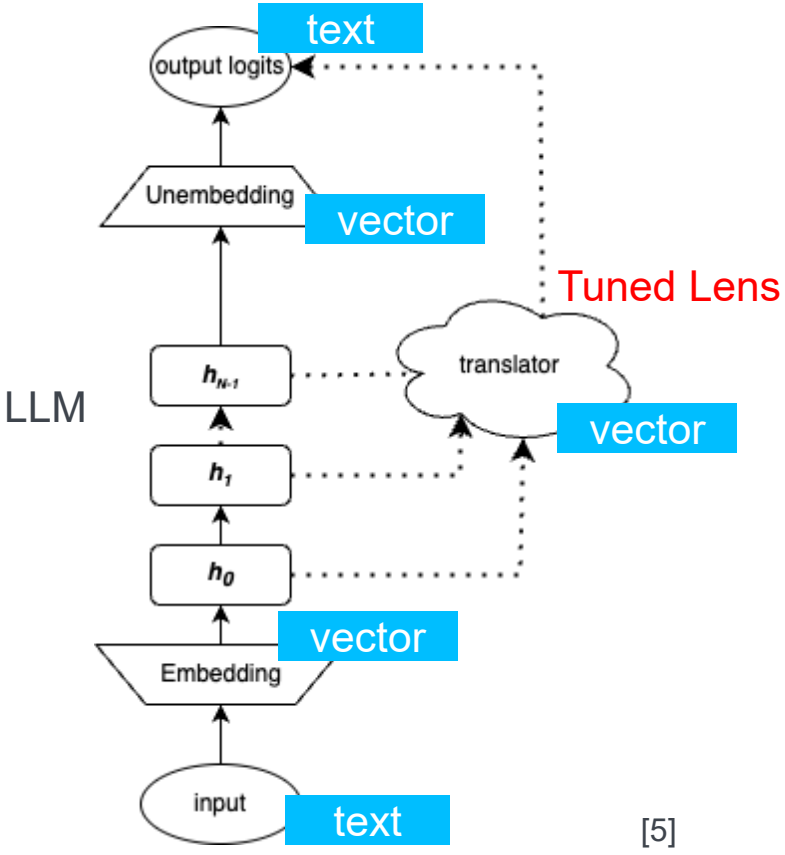
Probing Techniques – Logit Lens

- **Flow:** Transforms input to embeddings, processes through layers to output logits, and probes its intermediate results.
- **Probing Intermediate States:** Directly using the model‘s unembedding layer to decode each hidden layer‘s attention mechanism output and block output.
  To understand how information evolves and contributes to the final output. [4]



Motivation & Basis

Probing Techniques – Tuned Lens



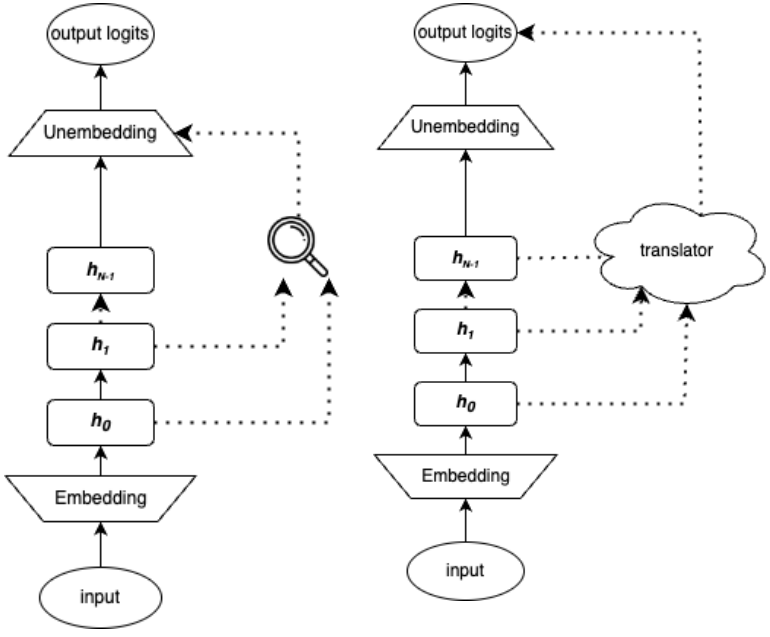
[5]

- Introduced in "**Eliciting Latent Predictions from Transformers with the Tuned Lens**"
- Similar to the **Logit Lens** implementation.
- The Tuned Lens refines the Logit Lens by introducing "translators," which use machine learning to train.
- These translators adjust the hidden states layer by layer, ensuring they more closely match those expected at the final output layer.

Conceptual Design

Comparison of Logit Lens and Tuned Lens

	Logit Lens (Left)	Tuned Lens (Right)
Lens Architecture	Projection	Transformation
Interpretability	Direct	Refined Token
Methodology	Mapping	Training (Machine Learning)
Transparency	High and Straightforward	High but close aligned with the final output

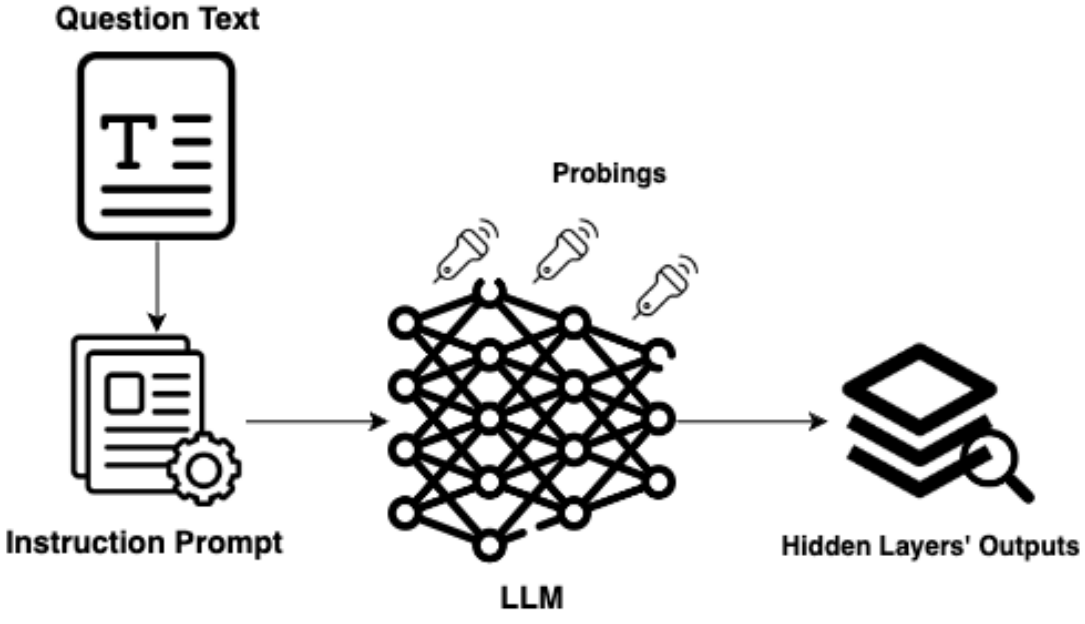


Conceptual Design

Conceptual Design

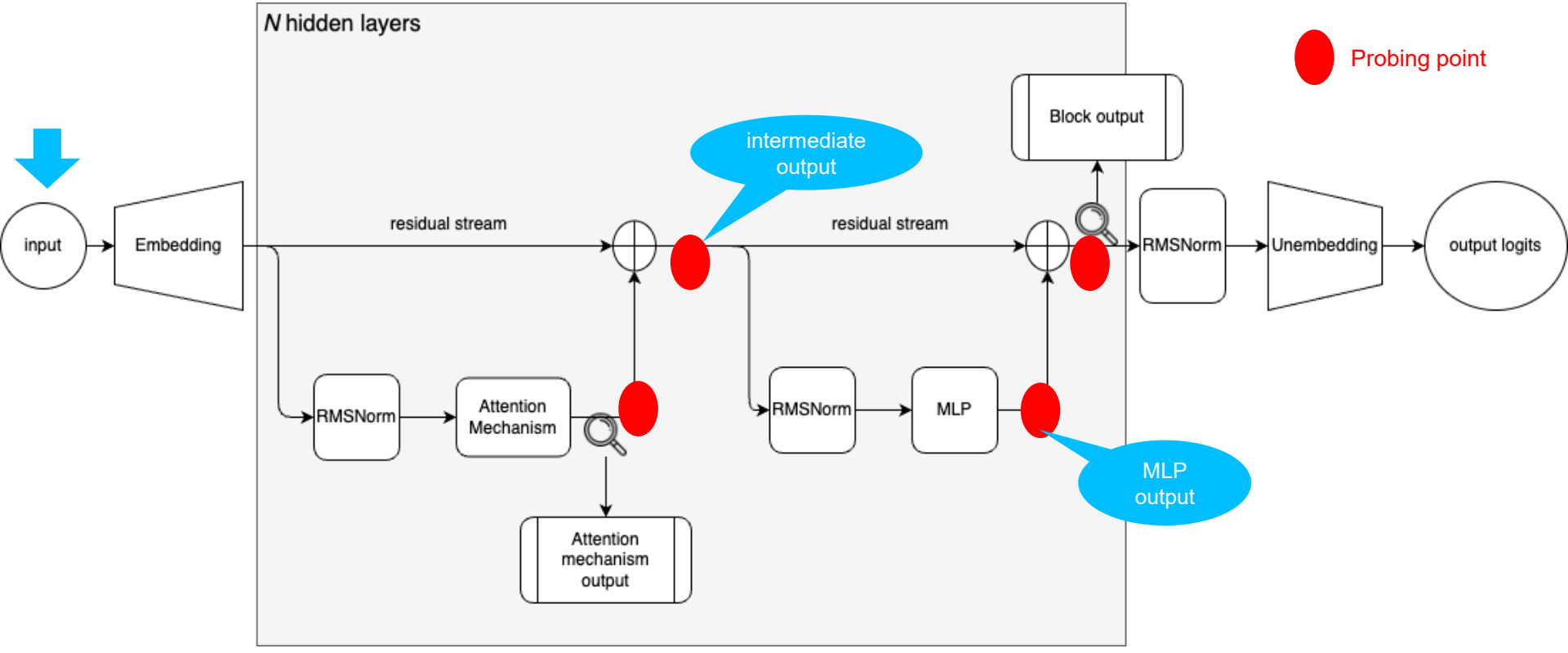
System Overview

- Instruction Prompt + Question Text
- Probing Technique on LLM's Hidden Layers
  - Logit Lens
  - Tuned Lens
- Analyzing the Hidden Layers' Outputs and make further processing.



Conceptual Design

Probing Technique in LlaMA 2 architecture



Implementation

Implementation

Instruction Prompt with Question Text

```
"""<s>[INST] <<SYS>>
Role and goal:
Your role is to act as an intelligent problem-solver, tasked with selecting the correct answer from a set of
multiple-choice options. Your goal is to analyze the question carefully and each of the provided options, applying
your extensive knowledge base and reasoning skills to determine the most accurate and appropriate answer.

Context:
The input text is a question with multiple-choice options. The correct answer is indicated by the option label A, B,
C, or D.
1. Question: A clear query requiring a specific answer.
2. Options: A list of possible answers labeled with possible answer A.First possible answer B.Second possible answer
C.Third possible answer D.Fourth possible answer

Instructions:
Analyze the question and options provided.
Use your knowledge to assess each option.
Employ reasoning to eliminate clearly incorrect options.
Identify the most accurate answer based on the information given.
Conclude by justifying your selection, clearly indicating your choice by referencing the option label A, B, C, or D.
You should only output one capitalized letter indicating the correct answer.

Example:
Input: // you will receive the question and options here.
Output: The correct answer is {one of A, B, C, D} // you will output the correct answer, replace {one of A, B, C, D}
with the correct option label A, B, C, or D.

Now you can start to answer the question with given options to give the correct answer.
<</SYS>>

Input: {{inputText}}[/INST]
Output: The correct answer is """
```

- Role and goal
- Context
- Instructions
- Example
- Input
- Output

Replace the “{{inputText}}” with the Question Text

Implementation

Question Text – Multiple Choices Question

Question: An industry that create a lot of waste is? Options: A.Solar Power Companies B.Recycling companies C.water production companies D.Cleaning Companies.

Question: Which of the following is NOT a characteristic of perfectly competitive industry? Options: A.Free entry into the industry B.Product differentiation C.Perfectly elastic demand curve D.Homogeneous products.

Question: Which of the following elements is not part of Porter's Five Forces model for industry competitiveness? Options: A.Threat of substitutes B.Threat of suppliers C.Power of buyers D.Threat from government.

Question: Which of the following is an example of a bulk-gaining industry? Options: A.Steel B.Bottled orange juice C.Paper D.Copper.

Question: The industry that makes plastic army figures uses a small fraction of the plastic demanded for all purposes. On this basis, we can conclude that the army-figures industry is most likely a(n)? Options: A.increasing-cost industry constant-cost industry B.decreasing-cost industry C.profit-making industry.

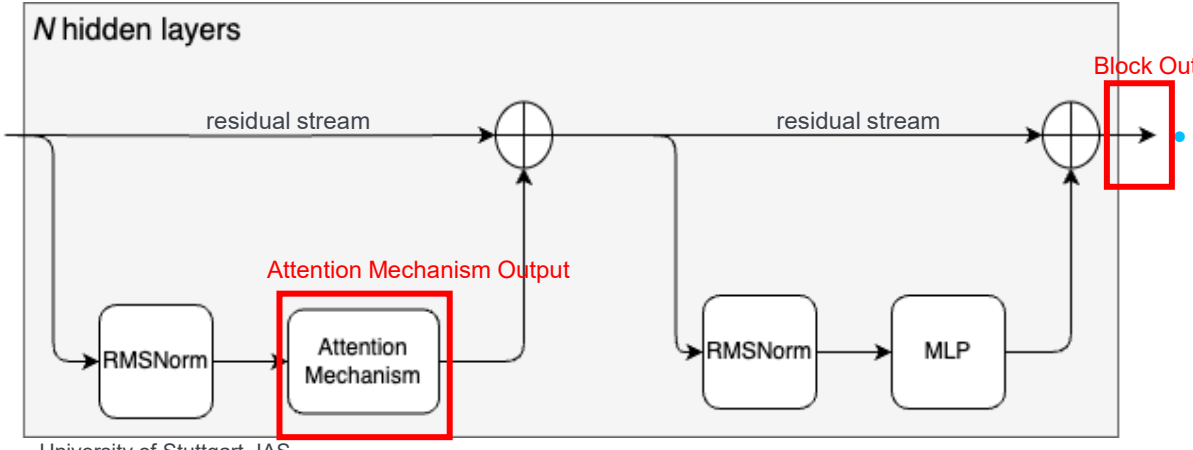
Question: Which of the following has the highest predictive validity in personnel selection in industry? Options: A.A projective technique B.An objective personality inventory C.An interview by the personnel manager D.A biographical inventory.

Implementation

Wrappers for Attention Mechanism and Block Outputs Interpretation

```
def forward(self, *args, **kwargs):
    output = self.block(*args, **kwargs)
    self.block_output = self.norm(output[0])
    attn_output = self.block.self_attn.activations
    self.attn_mech_output = self.norm(attn_output)
    attn_output += args[0]
    self.intermediate_res = self.norm(attn_output)
    mlp_output = self.block.mlp(self.post_attention_layernorm(attn_output))
    self.mlp_output = self.norm(mlp_output)
    return output
```

LLaMa 1 layer

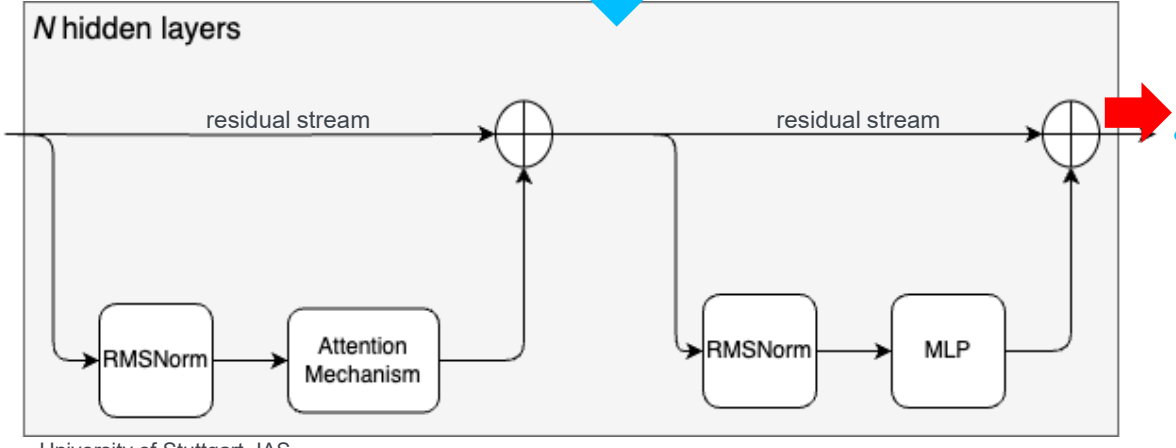


- The wrapper for input progresses through the model in order to decode the intermediate results.

Implementation

Wrappers for Attention Mechanism and Block Outputs Interpretation

```python
def forward(self, *args, **kwargs):
    output = self.block(*args, **kwargs)
    self.block_output = self.norm(output[0])
    attn_output = self.block.self_attn.activations
    self.attn_mech_output = self.norm(attn_output)
    attn_output += args[0]
    self.intermediate_res = self.norm(attn_output)
    mlp_output = self.block.mlp(self.post_attention_layernorm(attn_output))
    self.mlp_output = self.norm(mlp_output)
    return output
```

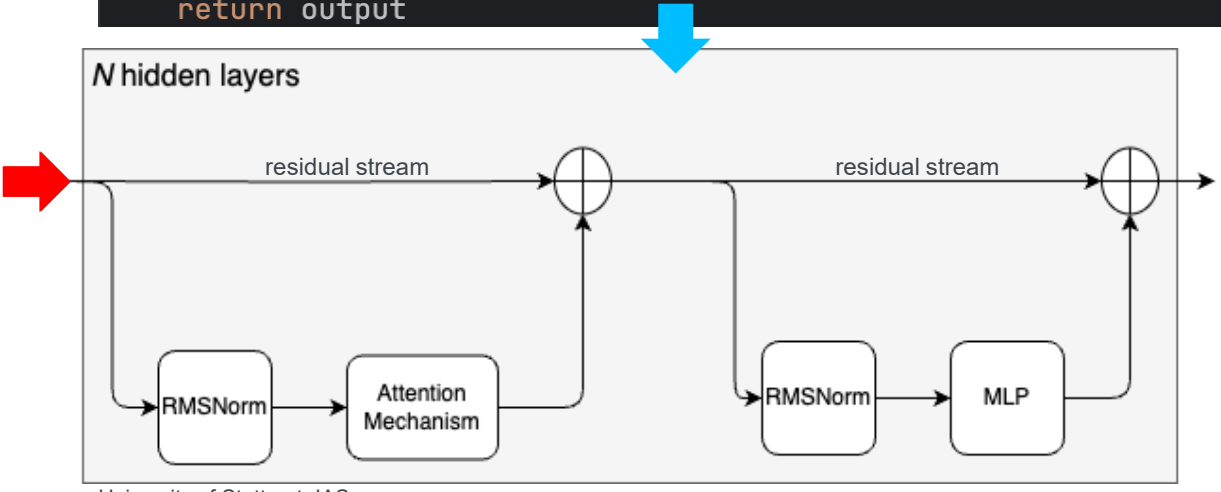


- The wrapper for input progresses through the model in order to decode the intermediate results.

Implementation

Wrappers for Attention Mechanism and Block Outputs Interpretation

```
def forward(self, *args, **kwargs):
    output = self.block(*args, **kwargs)
    self.block_output = self.norm(output[0])
    attn_output = self.block.self_attn.activations
    self.attn_mech_output = self.norm(attn_output)
    attn_output += args[0]
    self.intermediate_res = self.norm(attn_output)
    mlp_output = self.block.mlp(self.post_attention_layernorm(attn_output))
    self.mlp_output = self.norm(mlp_output)
    return output
```

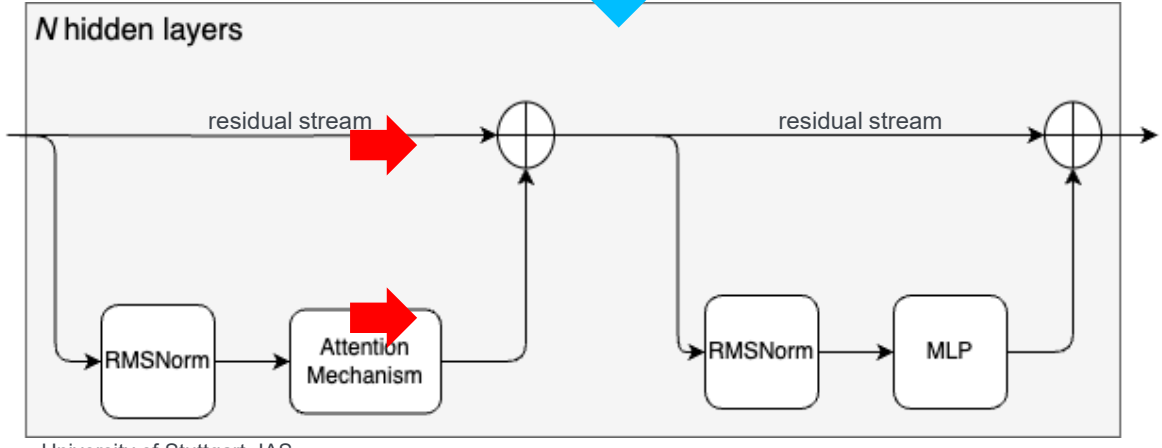


- The wrapper for input progresses through the model in order to decode the intermediate results.

Implementation

Wrappers for Attention Mechanism and Block Outputs Interpretation

```
def forward(self, *args, **kwargs):
    output = self.block(*args, **kwargs)
    self.block_output = self.norm(output[0])
    attn_output = self.block.self_attn.activations
    self.attn_mech_output = self.norm(attn_output)
    attn_output += args[0]
    self.intermediate_res = self.norm(attn_output)
    mlp_output = self.block.mlp(self.post_attention_layernorm(attn_output))
    self.mlp_output = self.norm(mlp_output)
    return output
```

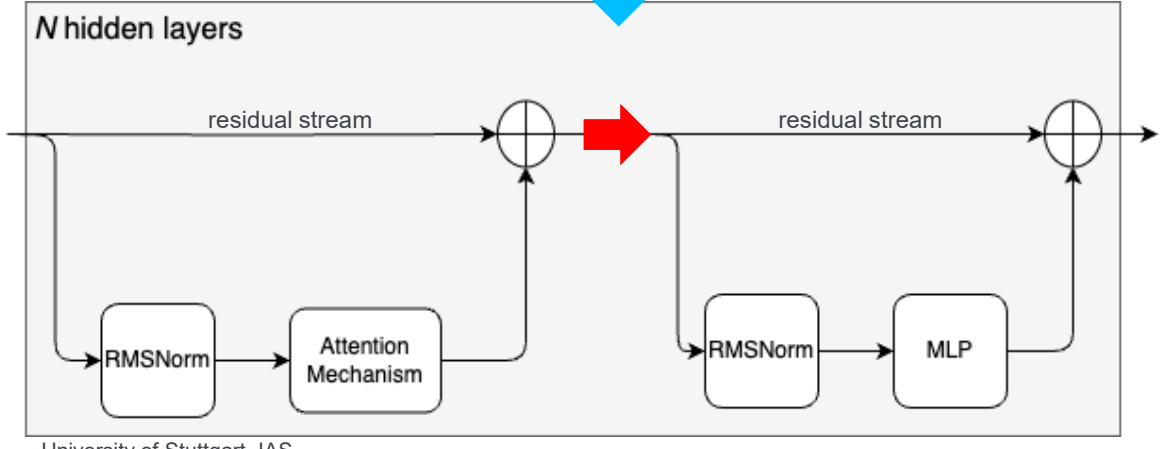


- The wrapper for input progresses through the model in order to decode the intermediate results.

Implementation

Wrappers for Attention Mechanism and Block Outputs Interpretation

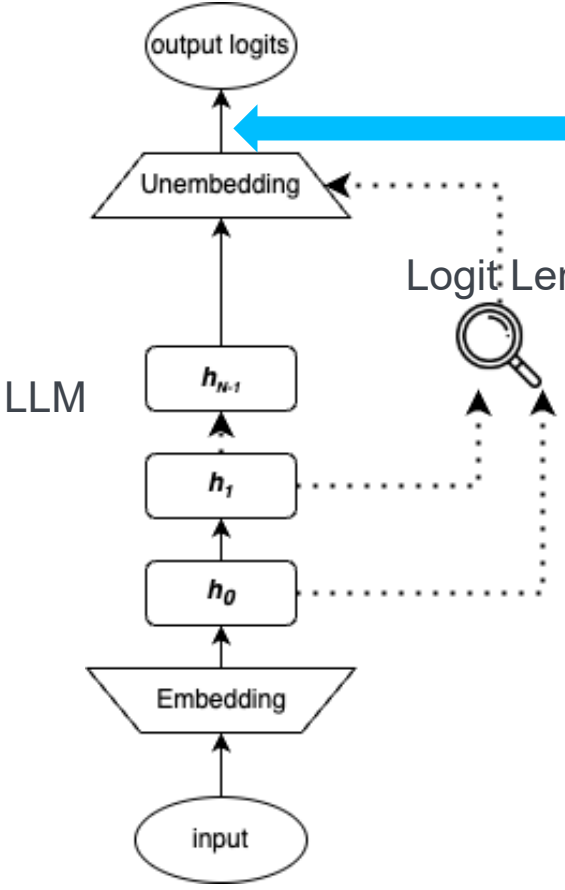
```
def forward(self, *args, **kwargs):
    output = self.block(*args, **kwargs)
    self.block_output = self.norm(output[0])
    attn_output = self.block.self_attn.activations
    self.attn_mech_output = self.norm(attn_output)
    attn_output += args[0]
    self.intermediate_res = self.norm(attn_output)
    mlp_output = self.block.mlp(self.post_attention_layernorm(attn_output))
    self.mlp_output = self.norm(mlp_output)
    return output
```



- The wrapper for input progresses through the model in order to decode the intermediate results.

Implementation

Logit Lens Interpretation



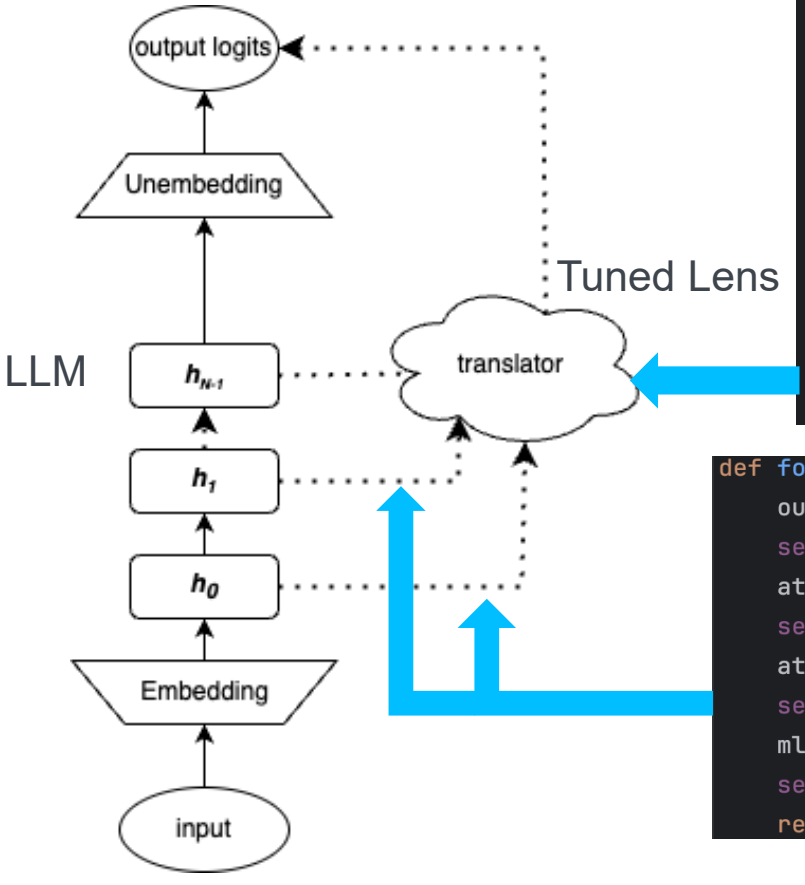
Logit Lens probing

```python
def print_decoded_activations(self, decoded_activations, label):
    softmaxed = torch.nn.functional.softmax(decoded_activations[0][-1], dim=-1)
    values, indices = torch.topk(softmaxed, k: 10)

    probs_percent = [round(v * 100, 2) for v in values.tolist()]
    tokens = self.tokenizer.batch_decode(indices.unsqueeze(-1))

    # Convert to list for CSV writing
    decoded_output = [label] + list(zip(tokens, probs_percent))
    return decoded_output
```

```python
def forward(self, *args, **kwargs):
    output = self.block(*args, **kwargs)
    self.block_output_unembedded = self.unembed_matrix(self.norm(output[0]))
    attn_output = self.block.self_attn.activations
    self.attn_mech_output_unembedded = self.unembed_matrix(self.norm(attn_output))
    attn_output += args[0]
    self.intermediate_res_unembedded = self.unembed_matrix(self.norm(attn_output))
    mlp_output = self.block.mlp(self.post_attention_layernorm(attn_output))
    self.mlp_output_unembedded = self.unembed_matrix(self.norm(mlp_output))
    return output
```

Implementation

Tuned Lens Interpretation




```python
def decode_and_print_top_tokens(self, hidden_states, layer_idx, block_output=False):
    # Transform hidden states into logits
    if block_output == False or layer_idx != 31:
        logits = self.tuned_lens.forward(hidden_states, layer_idx)
        probs = torch.nn.functional.softmax(logits[0][-1], dim=-1)
    else:
        probs = torch.nn.functional.softmax(self.model.lm_head(hidden_states[0][-1]), dim=-1)

    top_probs, top_indices = torch.topk(probs, k: 10, dim=-1)

    tokens = self.tokenizer.convert_ids_to_tokens(top_indices.squeeze().tolist())
    probabilities = top_probs.squeeze().tolist()
    probs_percent = [round(v * 100, 2) for v in probabilities]
    token_prob_pairs = list(zip(tokens, probs_percent))
    return token_prob_pairs
```

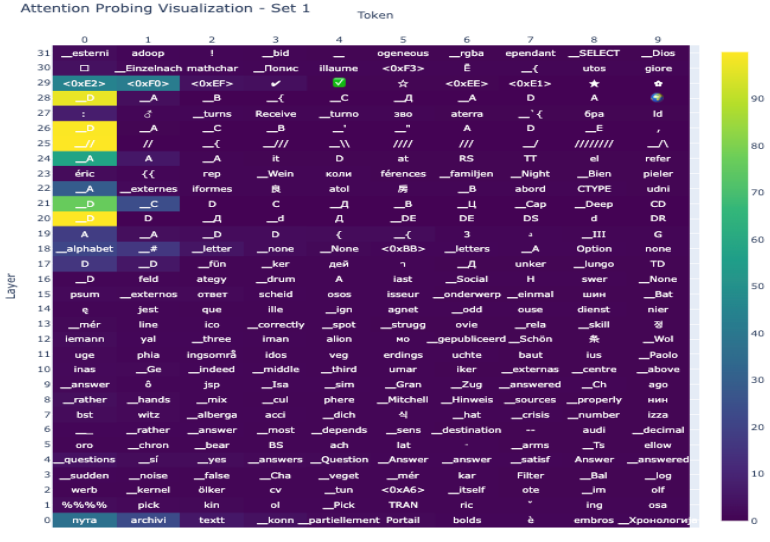
```python
def forward(self, *args, **kwargs):
    output = self.block(*args, **kwargs)
    self.block_output = self.norm(output[0])
    attn_output = self.block.self_attn.activations
    self.attn_mech_output = self.norm(attn_output)
    attn_output += args[0]
    self.intermediate_res = self.norm(attn_output)
    mlp_output = self.block.mlp(self.post_attention_layernorm(attn_output))
    self.mlp_output = self.norm(mlp_output)
    return output
```

Implementation

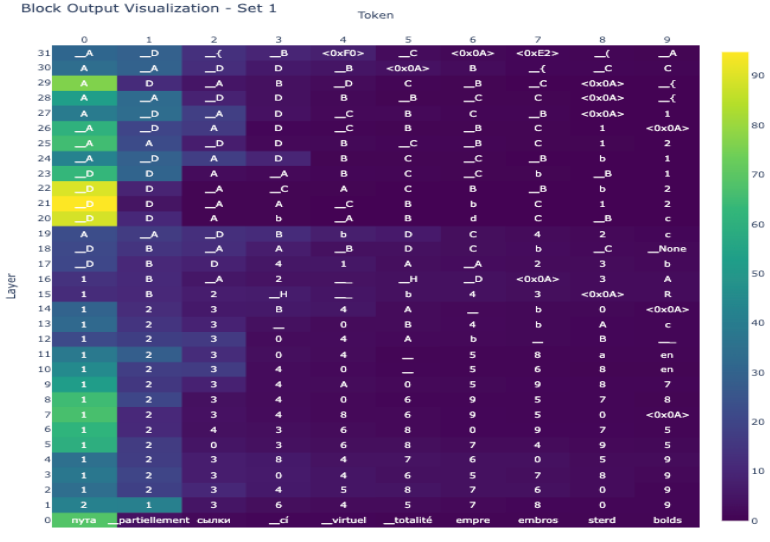
Visualizing the CSV data in Heatmaps

- Create interactive heatmaps based on each token's probabilities.

Layer	Attention mechanism
layer0	[('–ø—Ê—Ç–∞', 37.25), ('archivi', 22.86), ('textt', 5.72), (',ñÅkonn', 3.91), (',ñÅpartieller
layer1	[('%%%%%', 2.4), ('pick', 1.4), ('kin', 1.36), ('ol', 1.05), (',ñÅPick', 0.82), ('TRAN', 0.67), ('ri
layer2	[('werb', 0.74), (',ñÅkernel', 0.72), ('√∂lker', 0.68), ('cv', 0.53), (',ñÅtun', 0.51), ('<0xA6>'
layer3	[(',ñÅsudden', 1.18), (',ñÅnoise', 0.91), (',ñÅfalse', 0.74), (',ñÅCha', 0.64), (',ñÅveget', 0.6
layer4	[(',ñÅquestions', 4.22), (',ñÅs√ª', 2.34), (',ñÅyes', 2.19), (',ñÅanswers', 1.64), (',ñÅQues
layer5	[('oro', 0.69), (',ñÅchron', 0.68), (',ñÅbear', 0.68), ('BS', 0.62), ('ach', 0.52), ('lat', 0.5), ('·
layer6	[(',ñÅ_', 1.26), (',ñÅrather', 1.21), (',ñÅanswer', 1.16), (',ñÅmost', 0.53), (',ñÅdepends', 0
layer7	[('bst', 2.14), ('witz', 1.5), (',ñÅalberga', 0.96), ('acci', 0.91), (',ñÅdich', 0.65), ('Ïâù', 0.59
layer8	[(',ñÅrather', 3.15), (',ñÅhands', 1.66), (',ñÅmix', 0.81), (',ñÅcul', 0.7), ('phere', 0.67), (',ñ
layer9	[(',ñÅanswer', 1.4), ('√¥', 1.03), ('jsp', 0.97), (',ñÅlsa', 0.89), (',ñÅsim', 0.88), (',ñÅGran', (
layer10	[('inas', 1.94), (',ñÅGe', 1.42), (',ñÅindeed', 1.25), (',ñÅmiddle', 0.94), (',ñÅthird', 0.52), (
layer11	[('uge', 1.62), ('phia', 0.9), ('ingsomr√•', 0.83), ('idos', 0.7), ('veg', 0.69), ('erdings', 0.56)
layer12	[('iemann', 2.24), ('yal', 1.22), (',ñÅthree', 1.14), ('iman', 1.03), ('alion', 0.72), ('–º–æ', 0.(
layer13	[(',ñÅm√©r', 1.3), ('line', 0.87), ('ico', 0.8), (',ñÅcorrectly', 0.77), (',ñÅspot', 0.49), (',ñÅs
layer14	[('fô', 0.89), ('jest', 0.82), ('que', 0.58), ('ille', 0.58), (',ñÅign', 0.57), ('agnet', 0.49), (',ñÅo
layer15	[('psum', 1.75), (',ñÅexternos', 1.53), ('–æ—≤–µ—Ç', 1.52), ('scheid', 0.81), ('osos', 0
layer16	[(',ñÅD', 1.04), ('feld', 0.88), ('ategy', 0.64), (',ñÅdrum', 0.63), ('A', 0.59), ('iast', 0.52), (',ñ



Hovered on Attention Heatmap: Token: __Social, Layer: layer16, Confidence: 0.51



Hovered on Block Output Heatmap: Token: __D, Layer: layer16, Confidence: 3.69

Evaluation

Evaluation

Result Comparison of Logit Lens and Tuned Lens

	Logit Lens	Tuned Lens
Attention Mechanism	Both look similar on each layer's attention mechanism output. However	
	• The option labels begin (A, B, C, D) to be focused in **layers 17-20** • More unrecognized words or non-English words, difficult to interpret	• The option labels begin to be focused in **layers 16-18** • **More readable English words**
Block Output	• Random word outputs before layers 17 - 20 • **The option label outputs after layers 17 - 20**	• The option number(1, 2, etc.) outputs before layers 17 - 18 • **The option label outputs after layers 17 - 18**

Evaluation

Result Comparison of Logit Lens and Tuned Lens

	Logit Lens	Tuned Lens
Attention Mechanism	Both look similar on each layer's attention mechanism output. However	
	• The option labels begin (A, B, C, D) to be focused in **layers 17-20** • More unrecognized words or non-English words, difficult to interpret	• The option labels begin to be focused in **layers 16-18** • **More readable English words**
Block Output	• Random word outputs before layers 17 - 20 • **The option label outputs after layers 17 - 20**	• The option number(1, 2, etc.) outputs before layers 17 - 18 • **The option label outputs after layers 17 - 18**

Evaluation

Result Comparison of Logit Lens and Tuned Lens

	Logit Lens	Tuned Lens
Attention Mechanism	Both look similar on each layer's attention mechanism output. However	
	• The option labels begin (A, B, C, D) to be focused in **layers 17-20** • More unrecognized words or non-English words, difficult to interpret	• The option labels begin to be focused in **layers 16-18** • **More readable English words**
Block Output	• Random word outputs before layers 17 - 20 • **The option label outputs after layers 17 - 20**	• The option number(1, 2, etc.) outputs before layers 17 - 18 • **The option label outputs after layers 17 - 18**

Evaluation

Result Comparison of Logit Lens and Tuned Lens

	Logit Lens	Tuned Lens
Attention Mechanism	Both look similar on each layer's attention mechanism output. However	
	• The option labels begin (A, B, C, D) to be focused in **layers 17-20** • More unrecognized words or non-English words, difficult to interpret	• The option labels begin to be focused in **layers 16-18** • **More readable English words**
Block Output	• Random word outputs before layers 17 - 20 • **The option label outputs after layers 17 - 20**	• The option number(1, 2, etc.) outputs before layers 17 - 18 • **The option label outputs after layers 17 - 18**

Evaluation

Result Comparison of Logit Lens and Tuned Lens

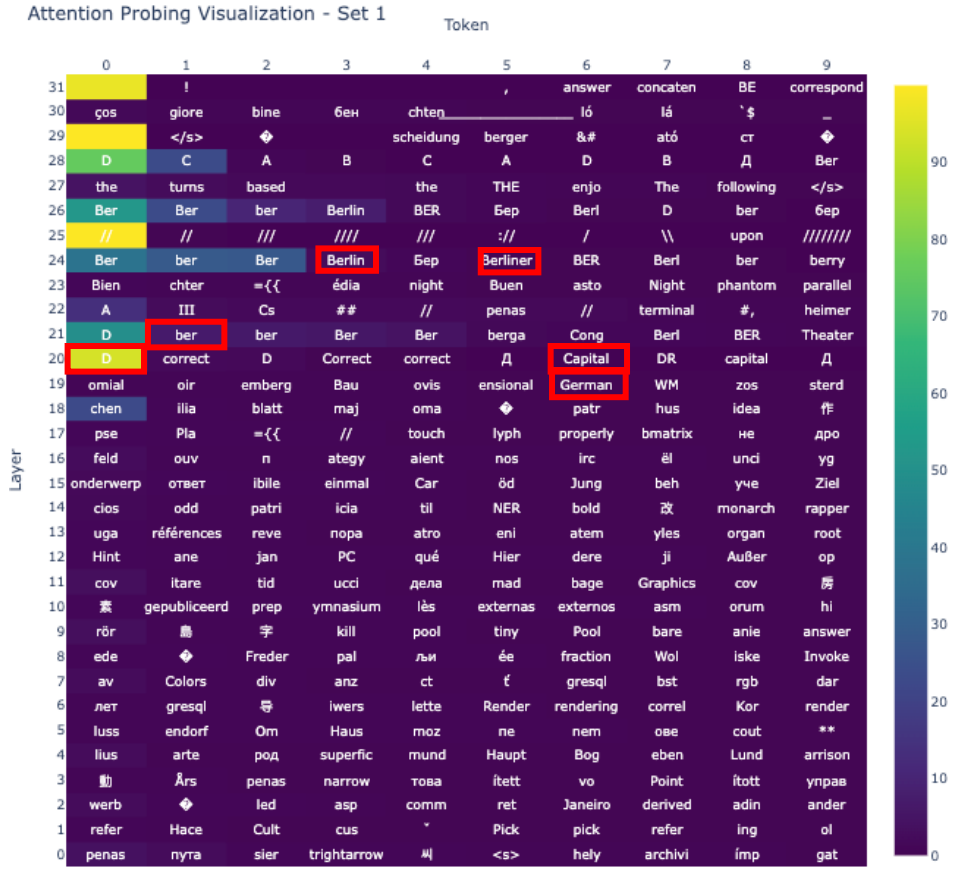
	Logit Lens	Tuned Lens
Attention Mechanism	Both look similar on each layer's attention mechanism output. However	
	• The option labels begin (A, B, C, D) to be focused in **layers 17-20** • More unrecognized words or non-English words, difficult to interpret	• The option labels begin to be focused in **layers 16-18** • **More readable English words**
Block Output	• Random word outputs before layers 17 - 20 • **The option label outputs after layers 17 - 20**	• The option number(1, 2, etc.) outputs before layers 17 - 18 • **The option label outputs after layers 17 - 18**

- Easier Question:
  Question: Which city is the capital of Germany?
  Options: A.London B.Paris C.Berlin D.Amsterdam

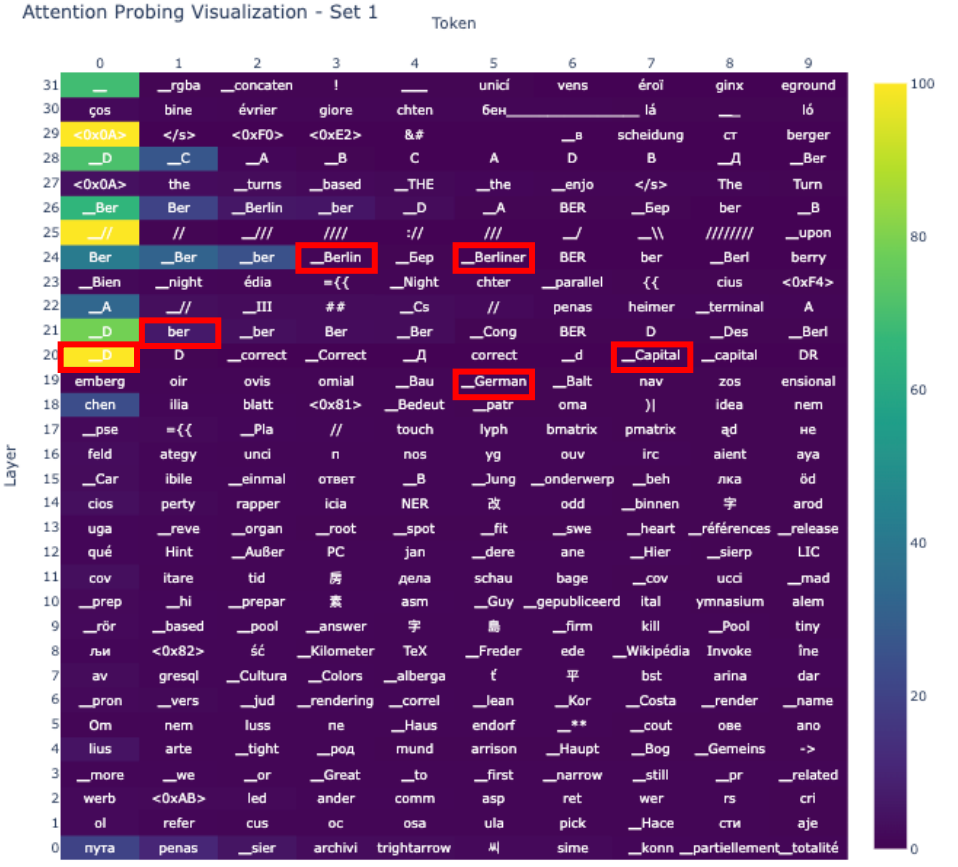
Evaluation

Attention Mechanism Output (Easier Question)

Question: Which city is the capital of Germany?
Options: A.London B.Paris C.Berlin D.Amsterdam



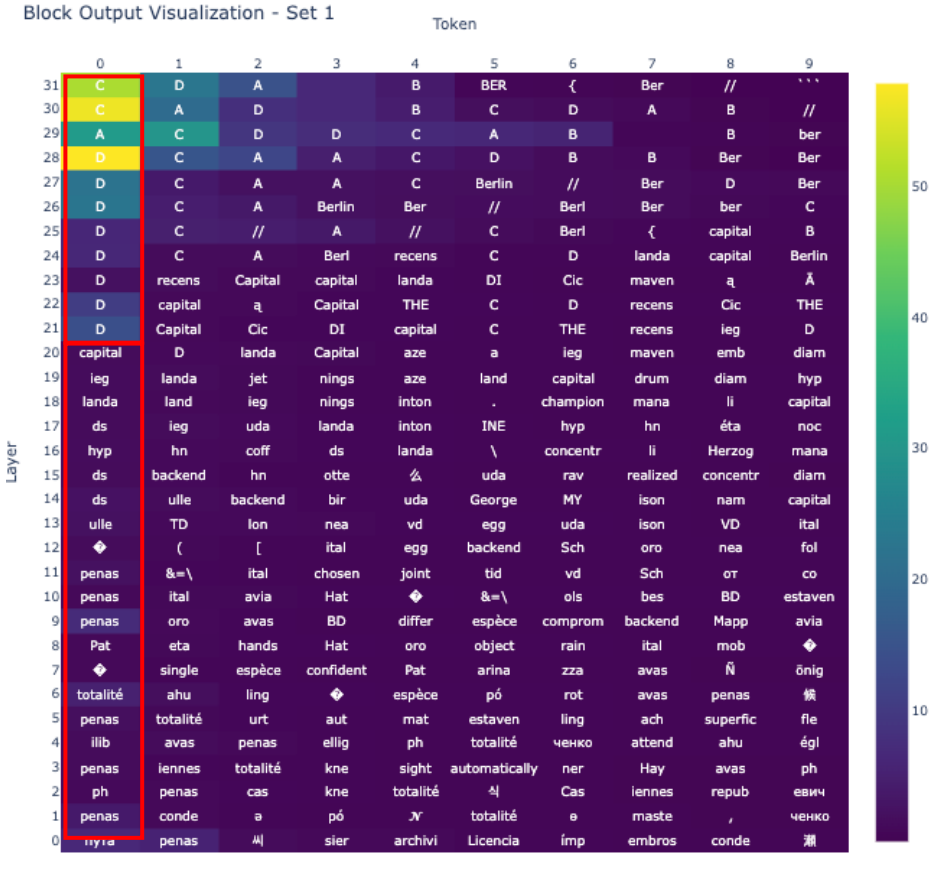
Logit Lens



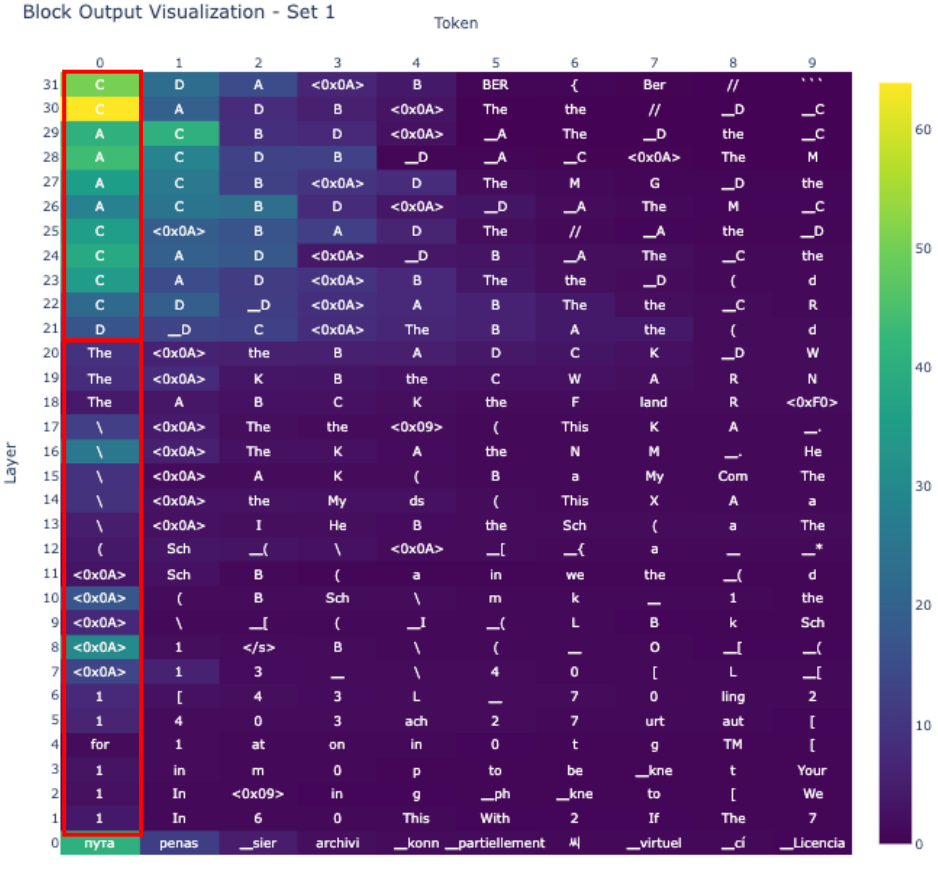
Tuned Lens

Evaluation

Block Output (Easier Question)



Logit Lens



Tuned Lens

Summary and Outlook

Summary and Outlook

- **Comparative Effectiveness in Model Layers**
  Attention Mechanism – Tuned Lens yields more predictive and understandable token interpretations than Logit Lens in complex prompts.
  Block Output – The performance of the Tuned Lens is more consistent across various hidden layers compared to that of the Logit Lens.

- **Training own Tuned Lens**
  The decoding result could be improved by **training own Tuned Lens** for different LLMs.

- **Limitations and Model Capacity**
  Llama-2-7B might not be able to handle complicated queries effectively.
  Upgrading to more robust models like Llama-2-13B or Llama-2-70B could enhance the accuracy of both intermediate and final results.

University of Stuttgart
Institut of Industrial Automation
and Software Engineering

Thank you!

Lun-Yu Yuan

e-mail st184762@stud.uni-stuttgart.de
phone +49 (0) 711 685-
fax +49 (0) 711 685-

University of Stuttgart
Institut of Industrial Automation and Software Engineering
Pfaffenwaldring 47, 70550 Stuttgart, Germany

Source

[1] https://pub.aimind.so/demystifying-the-black-box-interpretable-machine-learning-6388bef4aeca

[2] https://arxiv.org/pdf/1706.03762

[3] https://cameronrwolfe.substack.com/p/llama-2-from-the-ground-up

[4] https://www.lesswrong.com/posts/fJE6tscjGRPnK8C2C/decoding-intermediate-activations-in-llama-2-7b#Future_work

[5] https://arxiv.org/pdf/2303.08112