Project Work III. - 2025/26/1

# Decomposing the activations of Large Language Models

**Author:** Sándor Zsombor
**Supervisor:** Lukács András

# Introduction

- LLMs exploded in popularity
- They're still considered a black box by most
- Observation: Identical outputs in different languages if prompt is mirrored
- **Hypothesis**: Can the task and the language be separated throughout the token generation calculations?

*Important: In this project we only consider **causal transformer** models, and by „model" or „LLM" we refer to a neural network like that.*

# Before we begin

**Main source:**

## Separating Tongue from Thought

***Activation Patching Reveals Language-Agnostic Concept Representations in Transformers***

by C. Dumas, C. Wendler, V. Veselovsky, G. Monea, R. West (2024)

- We replicate and build on their results
- We utilize and take inspiration from techniques described

# Activation patching

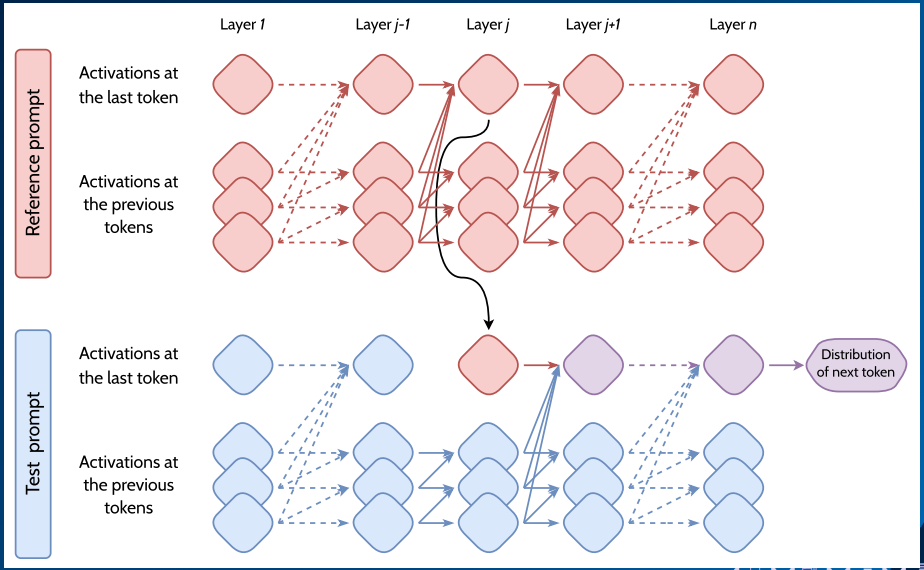
# Main technique

- **Activation patching**: Take an input, start computing the output with the model, and throughout the calculation, modify the activations „on the fly", with inserting „dirty activations" in place of calculated ones before continuing.
- We observe layered, transformer based LLMs, therefore
  - The input is given as a string („*prompt*"), and encoded by a tokenizer
  - The model has $n$ layers of equal dimensionality $d$
  - A layer calculates outputs a $d$-dimensional representation for each token

# The usual form

1. Take a causal language model, one or multiple reference prompts, a test prompt, and a layer index *j*.
2. **For each reference prompt**: calculate the next token with the LM, and extract the activation of the last token at the specified layer. At the end, average the reference activations, this will be our „mean reference activation at layer *j*" ($MRA_j$).
3. Execute the test prompt with patching applied at layer *j* to the activation of the last token with $MRA_j$ in the role of „dirty activation".
4. Observe the token distribution at the end of the test run.

# Activation patching, single reference prompt



# Problem setting

# Problem setting

- Single-word translation task
- Aim: separate the target language and the translated concept
- Dataset: Small sample of BabelNet – 123 concepts and 13 languages
- Prompt format: 5 shot, without any task description
- Model: Llama 3.1 8B
- $(L_s, C, L_t)$: the task of translating the concept $C$ from source language $L_s$ to target language $L_t$
- Source language is less important, therefore $(C, L)$ is the task of translating concept $C$ to language $L$

# Example prompt

*Deutsch: „Division" - Italiano: „divisione"*
*Deutsch: „Norden" - Italiano: „nord"*
*Deutsch: „Bekleidung" - Italiano: „vestiti"*
*Deutsch: „Maschine" - Italiano: „macchina"*
*Deutsch: „Abbild" - Italiano: „pittura"*
*Deutsch: „Herz" - Italiano: "*

# Central hypothesis

**Decomposition Hypothesis:**

1. *There is a layer with index $\ell_L$ at which the last token's activation* $\mathbf{x}$ *can be decomposed into 2 parts* $\mathbf{x} = \mathbf{x}_L + \mathbf{r}_L$ in a way that $\mathbf{x}_L$ fully controls the target language of the translation, and it does not interfere with the concept.
2. *There is a layer with index $\ell_C$ at which the last token's activation* $\mathbf{y}$ *can be decomposed into 2 parts* $\mathbf{y} = \mathbf{y}_C + \mathbf{r}_C$ in a way that $\mathbf{y}_C$ fully controls the concept of the translation, and it does not interfere with the target language.

**Subspatial Decomposition Hypothesis**: In addition, the $\mathbf{x}_L$ and $\mathbf{y}_C$ vectors span subspaces at their respective layers.

# Terminology

**„The model generates *C* in *L* with probability *p*”**:

Given the possible representations of the concept *C* in the language *L* (*the words in L that mean C*), sum the probabilities of their first tokens over the generated
token probability distribution.

# Experiment 0:
# Replicating the original results

# The original experiment and results

The experiment of Dumas et al.:

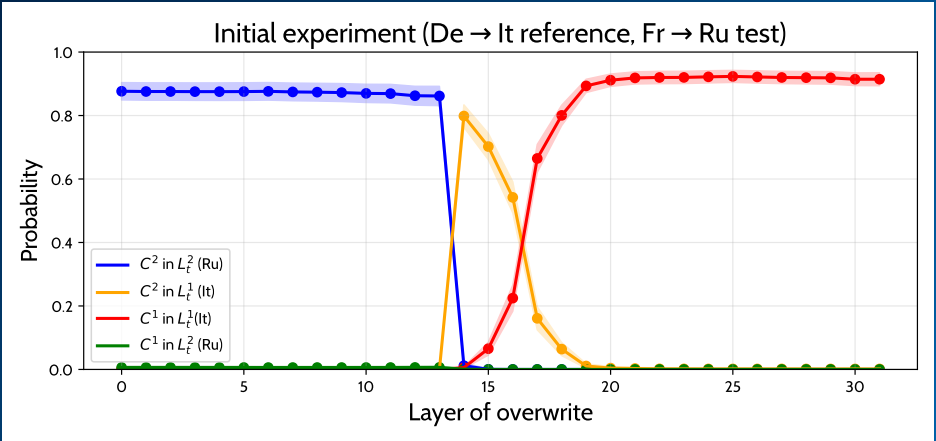
- Take 2 distinct translation tasks $(C^1, L^1)$ and $(C^2, L^2)$; their prompts: $\mathcal{P}^1$, $\mathcal{P}^2$.
- Apply activation patching to each layer (in a separate run), with $\mathcal{P}^1$ as reference and $\mathcal{P}^2$ as test prompt.
- Record probabilities of generating $C^1$ and $C^2$ in $L^1$ and $L^2$.

Their results:

- Patching early: the model generates $C^2$ in $L^2$ with high probability.
- Patching in the middle: the model generates $C^2$ in $L^1$ with high probability.
- Patching early: the model generates $C^1$ in $L^1$ with high probability.

We're able to replicate these results!

# Confirmation of results



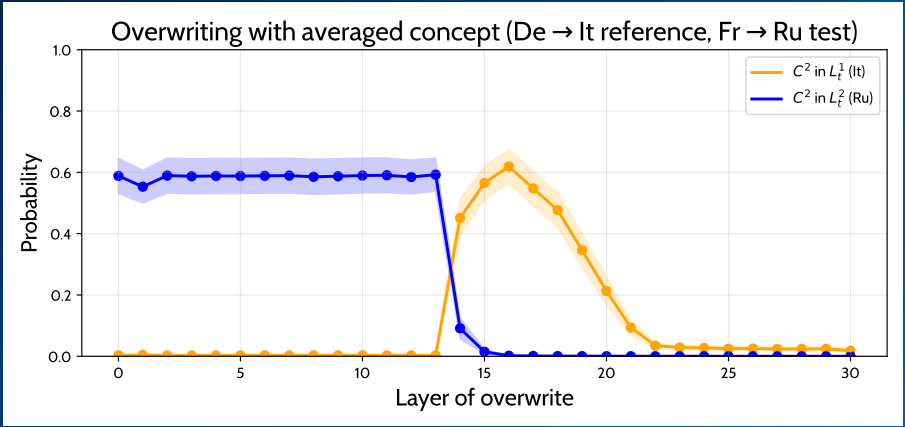

Probabilities of generating $C^1$ and $C^2$ in the target languages, averaged over 200 samples, depicting a 95% confidence interval.

# Experiment 1
# Decoupling the language

# Initial experiment

- Take $L^1$ language, and $(C_1^1, L^1), \cdots, (C_k^1, L^1)$ task (distinct). Take an $L^2$ language and $C^2$ concept.
- Apply activation patching to each layer (in a separate run), with $\mathcal{P}_1^1, \cdots, \mathcal{P}_k^1$ as reference prompts and $\mathcal{P}^2$ as test.
- Record the probabilities of generating $C^2$ in $L^1$ and $L^2$.

# Initial results

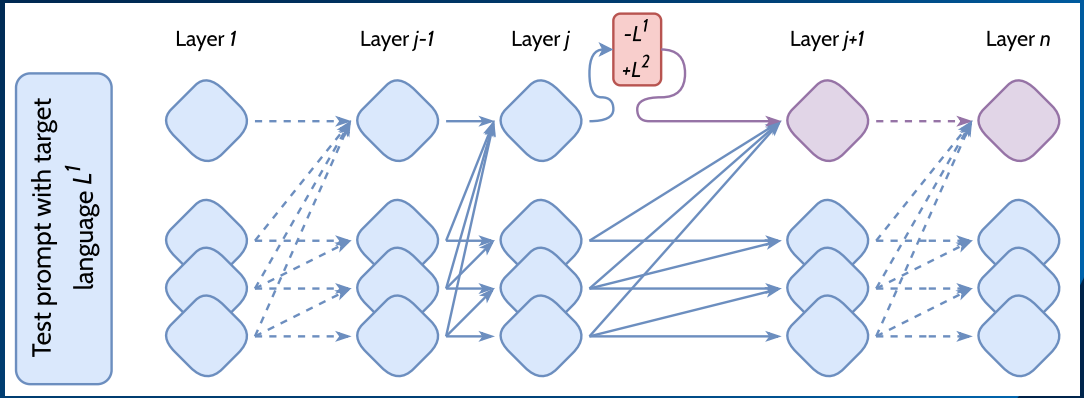



Probabilities of generating a given concept $C^2$ in the original language and the one patched. The figure shows reference translation German→ Italian and test translation French → Russian, with probabilities averaged over 93 samples, depicting a 95% confidence interval.

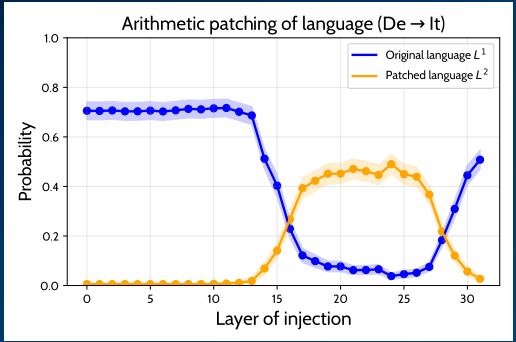
# Extracting vector representation

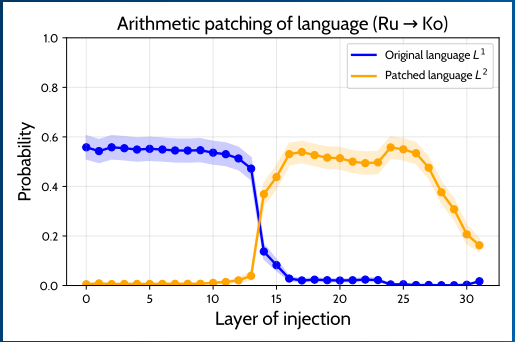
- Take experiment 1 with $L \equiv L_t^1$
- Let $\mathfrak{L}_i^L := \mathrm{MRA}_i$
- Hypothesis: These act as „components” in their respective layers, meaning we can use them in an arithmetic fashion

# Arithmetic injection



# Testing arithmetic injection



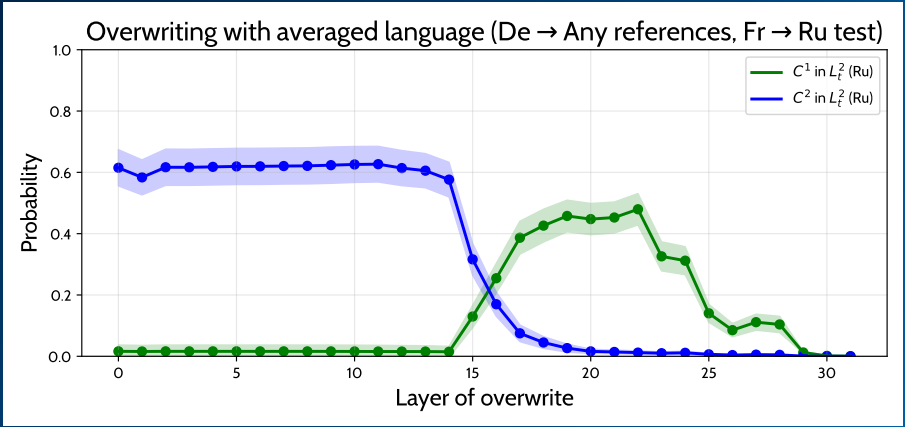

Probabilities of generating a given concept in the original language and the one injected arithmetically. The figures show translations German→ Italian and Russian→ Korean, with probabilities averaged over 200 samples, depicting a 95% confidence interval.

# Experiment 2
# Decoupling the concept

# Analogous results

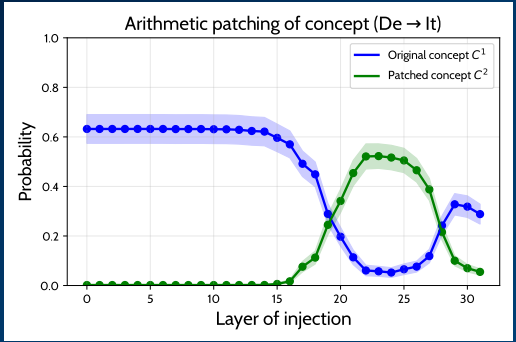
- Initial experiment is similar: instead of averaging over $(C_j^1, L^1)$, we average over $(L_{s_j}^1, C^1, L_{t_j}^1)$
- The result is similar, therefore we attempt to extract $\mathfrak{C}_i^C$ concept vectors
- Result: arithmetic injection can also be applied with concept vectors.

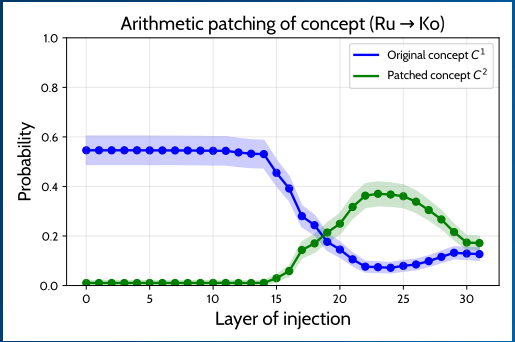
# Initial experiment




Probabilities of generating concepts $C^1$ and $C^2$ in the original language. The figure shows reference translation German→ Italian and test translation French→ Russian, with probabilities averaged over 93 samples, depicting a 95% confidence interval.

# Arithmetic injection of concept




Probabilities of generating the original concept and the one injected arithmetically in a given language. The figures show translations German→ Italian and Russian→ Korean, with probabilities averaged over 200 samples, depicting a 95% confidence interval.

# Experiment 3
# Complete redirection

# Combining redirections

Problem:

- Patching multiple times at the last token overwrites previous progress.
- $\Rightarrow$ Cannot hope to modify both language and concept independently with normal patching.

Solution: Arithmetic injection

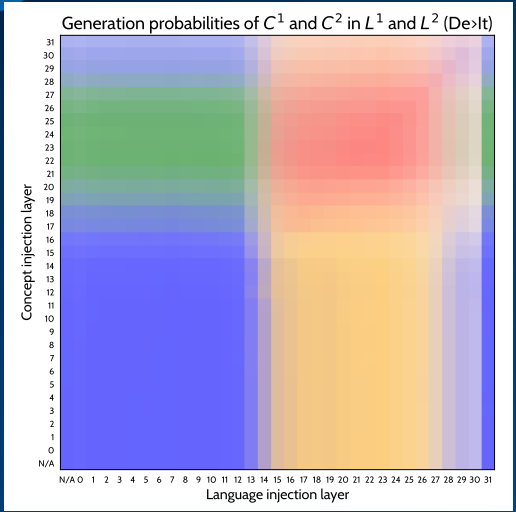
- Arithmetic injection does not overwrite activation
- $\Rightarrow$ Multiple injections are possible

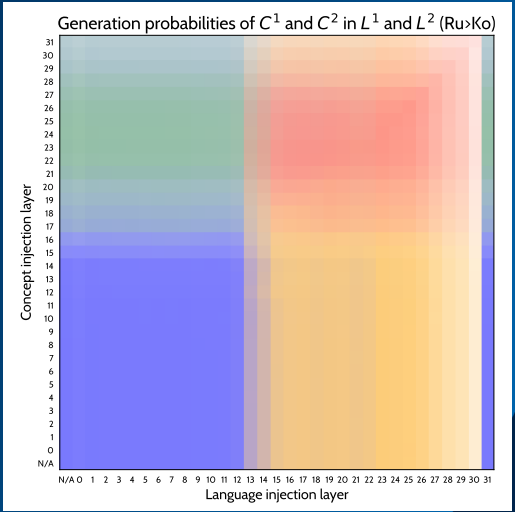
**Experiment**: Can we inject a concept and a language at the same time?

**Result**: Yes, it is possible!

**Corollary: The Decomposition Hypothesis stands!** *(in some form)*

# Results


Generation probabilities of $C^1$ and $C^2$ in $L^1$ and $L^2$ (De>It)
Concept injection layer
N/A 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Language injection layer


Generation probabilities of $C^1$ and $C^2$ in $L^1$ and $L^2$ (Ru>Ko)
Concept injection layer
N/A 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Language injection layer

# The shape of decompositions

# Hypothesis and methodology

Focus:

- We explore the Subspatial Decomposition Hypothesis
- Aim: Find the subspaces of the $\mathfrak{L}$ and $\mathfrak{C}$ vectors
- Note: Low number of samples $\Rightarrow$ limited results expected

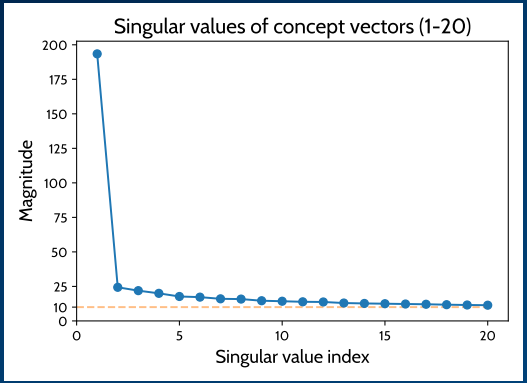
Methodology:

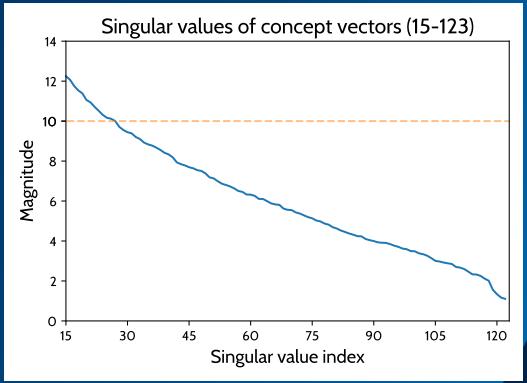
- SVD on the vectors
- Project to the singular vectors
- Observe the disposition for the first few projections

# Results

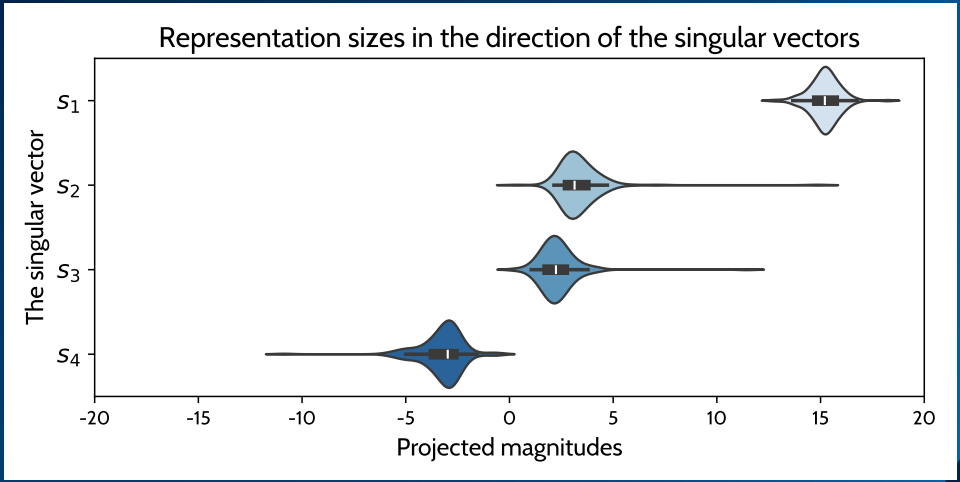
- No clear subspatial alignment
- First singular vector is strong, but others are also prominent
- Representations seem **clustered in an ellipsoid**
- Note: No significant changes in results when data is centered

# Singular values of the concept vectors

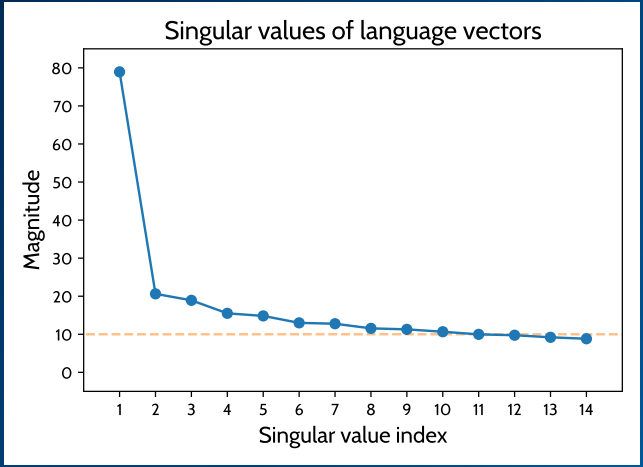


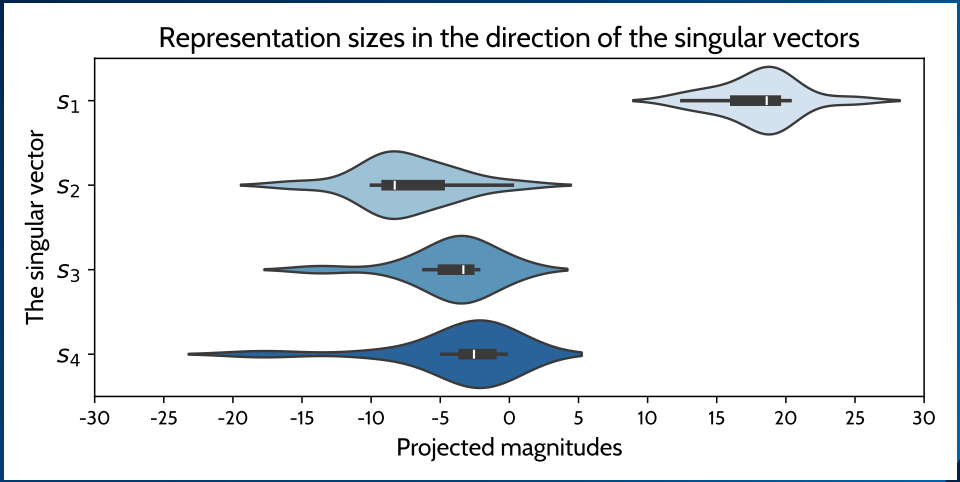
# Concept vector projected sizes



# Singular values of the language vectors



# Language vector projected sizes



# Appendix: AI usage

- The project is focused on LLM research, for which the Llama 3.1 8B model was used on the infrastructure of ELTE
- Claude Code was utilized for creating the initial codebase of the experiments
- ChatGPT aided further code modifications, debugging, and the generation of plots