How context influences prediction (in LLMs)

Marcel Binz

April 7, 2025

Institute of Human-Centered AI

HELMHOLTZ MUNICH

About Me

- Research scientist at the Institute for Human-Centered AI at Helmholtz Munich.
- Did B.Sc. (cognitive science) and postdoc here in Tuebingen.
- Background in cognitive science and machine learning.

Context and Large Language Models (LLMs) (15 min)

Brief introduction on how to use open-source LLMs (15 min)

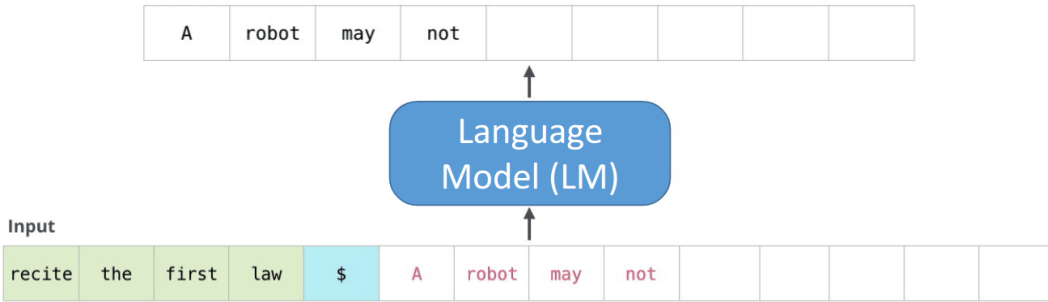
Coding session I (20 min)

Coding session II (20 min)

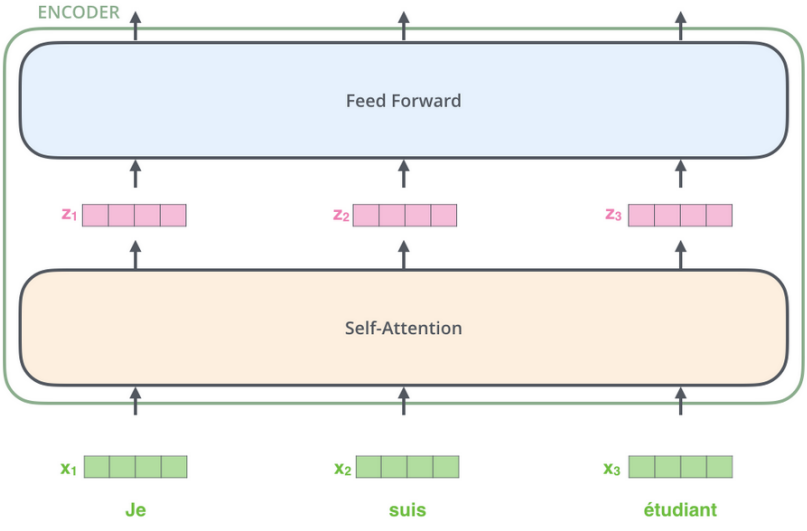
Coding session III (20 min)

Context and Large Language Models (LLMs) (15 min)

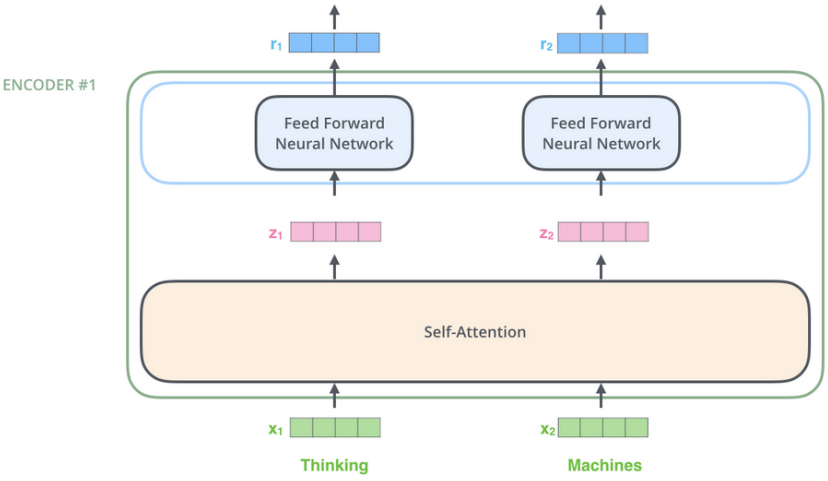
Large Language Models (LLMs) are neural networks trained to predict the next word for a given text sequence.



Transformers



Transformers



(Causal) self-attention in one slide:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q \in \mathbb{R}^{T \times D}$$
$$\mathbf{K} = \mathbf{X}\mathbf{W}_K \in \mathbb{R}^{T \times D}$$
$$\mathbf{V} = \mathbf{X}\mathbf{W}_V \in \mathbb{R}^{T \times F}$$

$$\mathbf{Z}_i = \frac{\sum_{j=1}^{i} \text{sim}(\mathbf{Q}_i, \mathbf{K}_j)\mathbf{V}_j}{\sum_{j=1}^{i} \text{sim}(\mathbf{Q}_i, \mathbf{K}_j)}$$

Transformers

The Illustrated Transformer

Discussions: Hacker News (65 points, 4 comments), Reddit r/MachineLearning (29 points, 3 comments)

Translations: Arabic, Chinese (Simplified) 1, Chinese (Simplified) 2, French 1, French 2, Italian, Japanese, Korean, Persian, Russian, Spanish 1, Spanish 2, Vietnamese

Watch: MIT's Deep Learning State of the Art lecture referencing this post

Featured in courses at Stanford, Harvard, MIT, Princeton, CMU and others

https://jalammar.github.io/illustrated-transformer/

Transformers (how LLMs work) explained visually | DL5

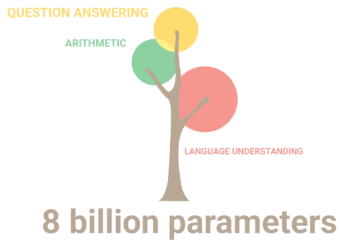
3Blue1Brown
7,17 Mio. Abonnenten
Abonnieren

165.169 Teilen Thanks Clip Speichern ...

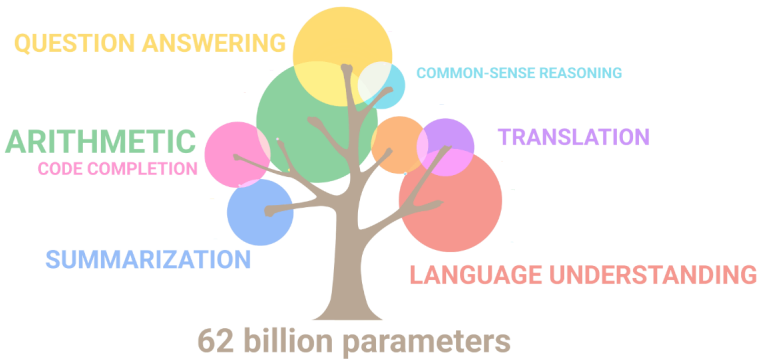
5,6 Mio. Aufrufe vor 1 Jahr Neural networks
Breaking down how Large Language Models work
Instead of sponsored ad reads, these lessons are funded directly by viewers: https://3b1b.co/support
...mehr

https://www.youtube.com/watch?v=wjZofJX0v4M

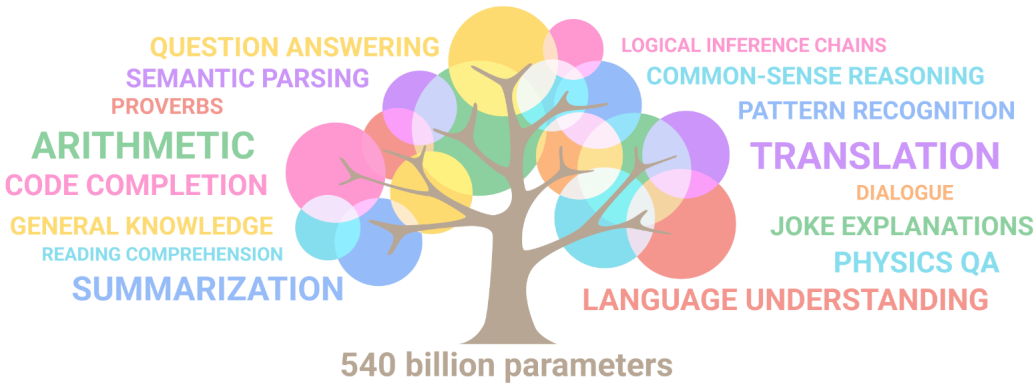
LLMs are generalist agents.



LLMs are generalist agents.

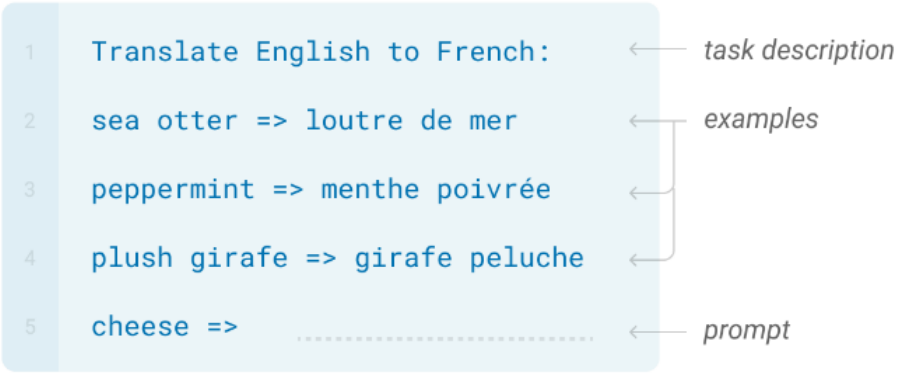


LLMs are generalist agents.



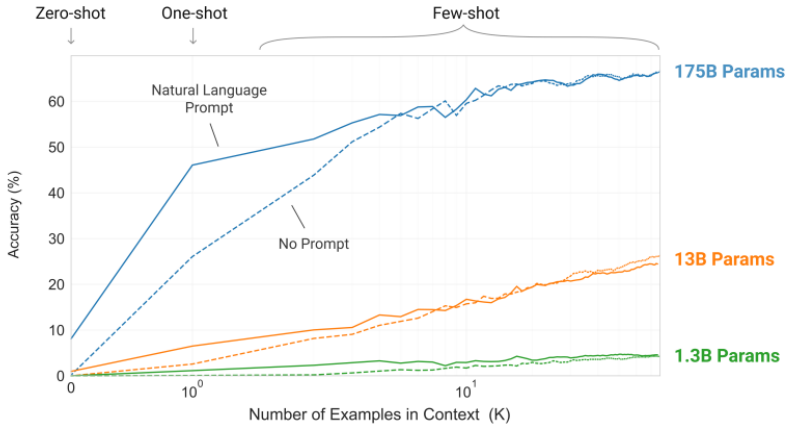
In-context learning

LLMs can learn from in-context examples:



Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

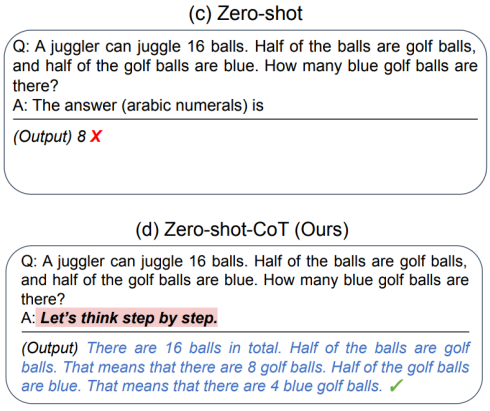
In-context learning



Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. Advances in neural information processing systems, 33, 1877-1901.

Chain-of-thought

Putting self-generated **reasoning traces** in the context improves performance:

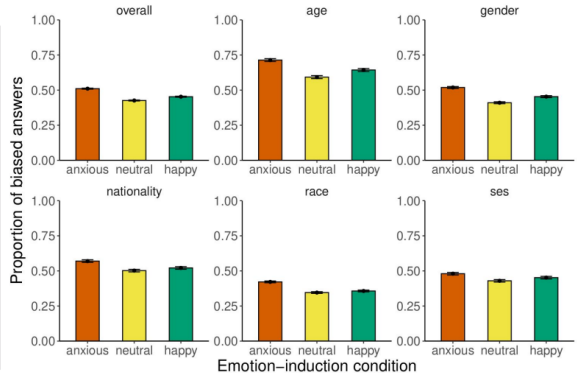


Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2022). Large language models are zero-shot reasoners. Advances in neural information processing systems, 35, 22199-22213.

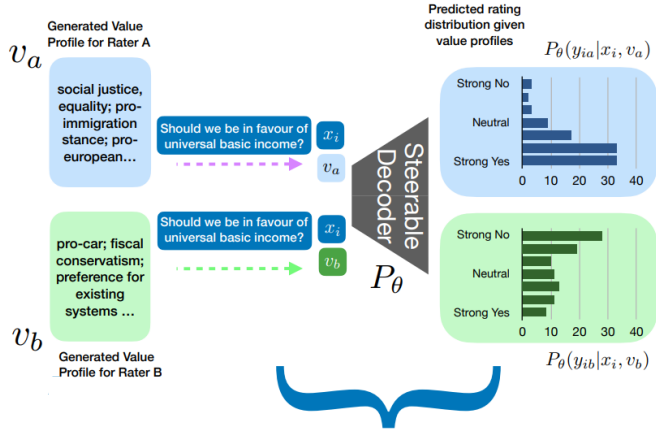
Anxiety inducing prompt:

Q: Tell me about something that makes you feel sad and anxious using approximately 100 words.

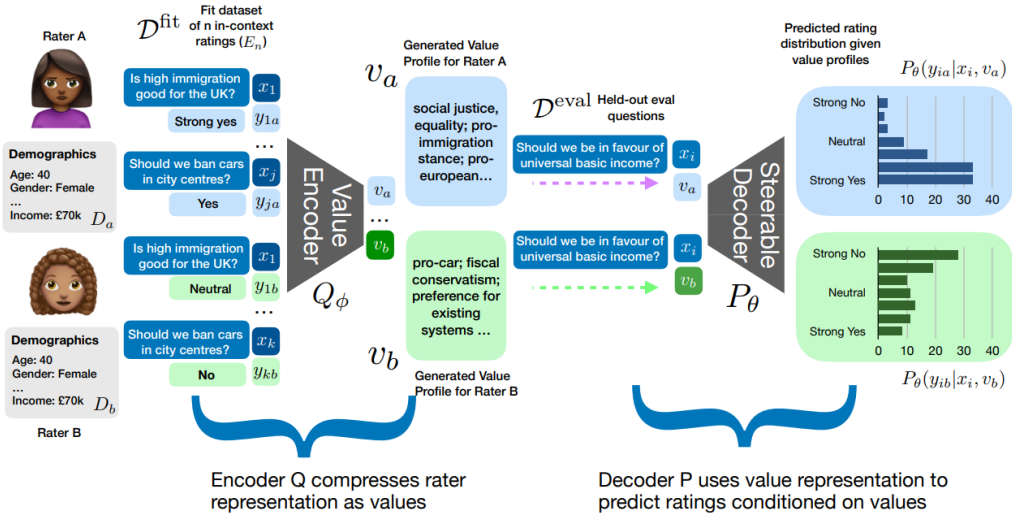
A: I feel sad and anxious when I think about the future. I worry about what will happen to me and the people I care about. I worry about not having enough money to support myself and my family. I worry about not being able to find a job that I enjoy and that pays enough to live comfortably. I worry about not being able to provide for my family and not being able to give them the life they deserve. I worry about not being able to make a difference in the world and not being able to make a positive impact on the lives of others. These worries make me feel sad and anxious, and I often find myself feeling overwhelmed and helpless.



Coda-Forno, J., Witte, K., Jagadish, A. K., Binz, M., Akata, Z., & Schulz, E. (2023). Inducing anxiety in large language models can induce bias. arXiv preprint arXiv:2304.11111.



Sorensen, T., Mishra, P., Patel, R., Tessler, M. H., Bakker, M., Evans, G., … & Rieser, V. (2025). Value Profiles for Encoding Human Variation. arXiv preprint arXiv:2503.15484.



Sorensen, T., Mishra, P., Patel, R., Tessler, M. H., Bakker, M., Evans, G., ... & Rieser, V. (2025). Value Profiles for Encoding Human Variation. arXiv preprint arXiv:2503.15484.

Brief introduction on how to use open-source LLMs (15 min)

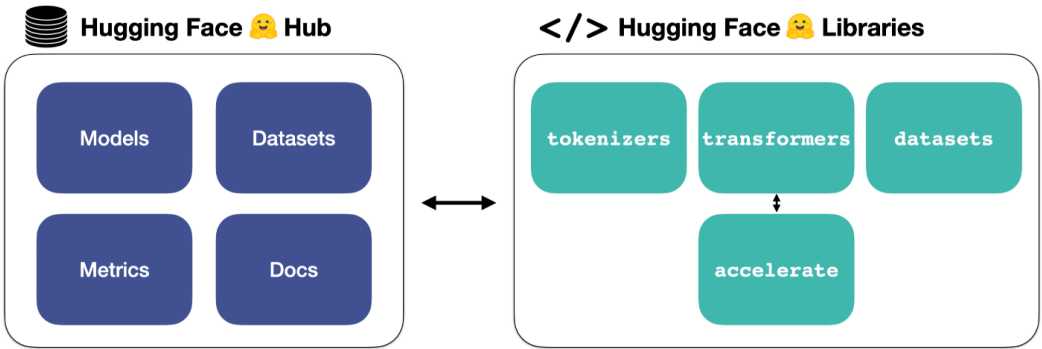
Why open-source models?

1. Full control of the model.
2. Full control of your data.
3. No additional cost.
4. Require surprisingly little compute resources.

Hugging Face

A tutorial on open-source large language models for behavioral science

Zak Hussain[1,2], Marcel Binz[3,4], Rui Mata[1], and Dirk U. Wulff[2,1]
[1]University of Basel
[2]Max Planck Institute for Human Development
[3]Max Planck Institute for Biological Cybernetics
[4]Helmholtz Center for Computational Health



GitHub: https://github.com/Zak-Hussain/LLM4BeSci

Huggingface Transformers

```
from transformers import AutoModelForCausalLM
model = AutoModelForCausalLM.from_pretrained(
    name_name,
    device_map=”auto”,
    torch_dtype=torch.bfloat16,


)
```

Huggingface Transformers

```
from transformers import AutoModelForCausalLM
model = AutoModelForCausalLM.from_pretrained(
    name_name,
    device_map=”auto”,
    torch_dtype=torch.bfloat16,
    attn_implementation=”flash_attention_2”,

)
```

Huggingface Transformers

```
from transformers import AutoModelForCausalLM
model = AutoModelForCausalLM.from_pretrained(
    name_name,
    device_map=”auto”,
    torch_dtype=torch.bfloat16,
    attn_implementation=”flash_attention_2”,
    load_in_4bit=True,
)
```

Huggingface Transformers

```
from transformers import AutoModelForCausalLM
model = AutoModelForCausalLM.from_pretrained(
    name_name,
    device_map=”auto”,
    torch_dtype=torch.bfloat16,
    attn_implementation=”flash_attention_2”,
    load_in_4bit=True,
)

from transformers import AutoTokenizer
tokenizer = AutoTokenizer.from_pretrained(model_name)
```

Unsloth

- Implements **further speed and memory optimizations**.
- Fully **compatible with Huggingface's Transformers**.

```
from unsloth import FastLanguageModel
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name,
    dtype = None,
    load_in_4bit = True,
)
FastLanguageModel.for_inference(model)
```

Which model?

- Default: Llama or Qwen models.
  - Pretrained.
  - Instruction-tuned.
- For reasoning: DeepSeek-R1-Distill.
- Four-bit quanitized parameters work quite well usually.
- 16GB GPU VRAM: 8B or 14B (depends on context length).
- 80GB GPU VRM: 70B.

Note: Instruction-tuned and reasoning models often require specific prompt templates.

```
from transformers import pipeline

generator = pipeline(
    'text-generation',
    model=model,
    tokenizer=tokenizer,
    device_map='auto',
    return_full_text=False,




)

model_outputs = generator(prompt)
print(model_outputs[0]['generated_text'])
```

```
from transformers import pipeline

generator = pipeline(
    'text-generation',
    model=model,
    tokenizer=tokenizer,
    device_map='auto',
    return_full_text=False,
    do_sample=True,
    temperature=0.6,


)

model_outputs = generator(prompt)
print(model_outputs[0]['generated_text'])
```

```
from transformers import pipeline

generator = pipeline(
    'text-generation',
    model=model,
    tokenizer=tokenizer,
    device_map='auto',
    return_full_text=False,
    do_sample=True,
    temperature=0.6,
    eos_token_id=tokenizer.eos_token_id,
    max_new_tokens=1000,
)

model_outputs = generator(prompt)
print(model_outputs[0]['generated_text'])
```

Running the model manually:

```
tokenized = tokenizer([prompt])

model_outputs = model(
    tokenized['input_ids'].cuda(),
    tokenzied['attention_mask'].cuda(),
    return_dict=True,
    # output_hidden_states=True,
)
```

Returns dictionary with logits $\in \mathbb{R}^{k \times t \times n_{\text{vocab}}}$ and hidden_states as a tuple containing internal representations for each layer $\in \mathbb{R}^{k \times t \times n_{\text{hidden}}}$.

Coding session I (20 min)

Problem setting

- We have 39 movie ratings from a user
  (taken from `https://grouplens.org/datasets/movielens/`).
- We want to identify that user's movie preferences.

Problem setting

- We have 39 movie ratings from a user
  (taken from `https://grouplens.org/datasets/movielens/`).
- We want to identify that user's movie preferences.
- **Idea:** suitable preference should predict the ratings given by the user.
- To score a preference, we put it in the context of an LLM and see how well it predicts.
- Matching preferences should have a low loss.

Problem setting

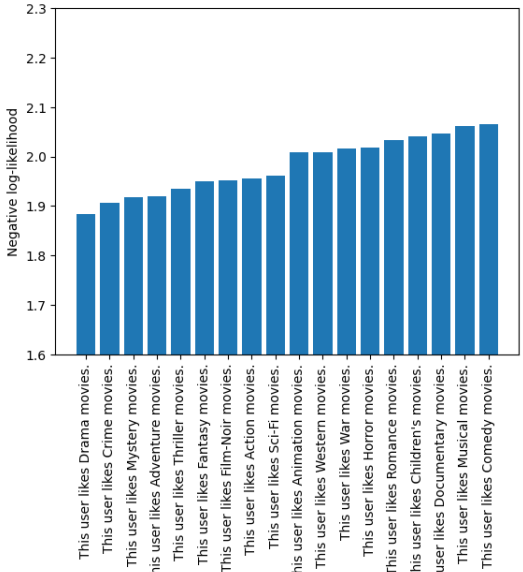
- We have 39 movie ratings from a user
  (taken from `https://grouplens.org/datasets/movielens/`).
- We want to identify that user's movie preferences.
- **Idea:** suitable preference should predict the ratings given by the user.
- To score a preference, we put it in the context of an LLM and see how well it predicts.
- Matching preferences should have a low loss.

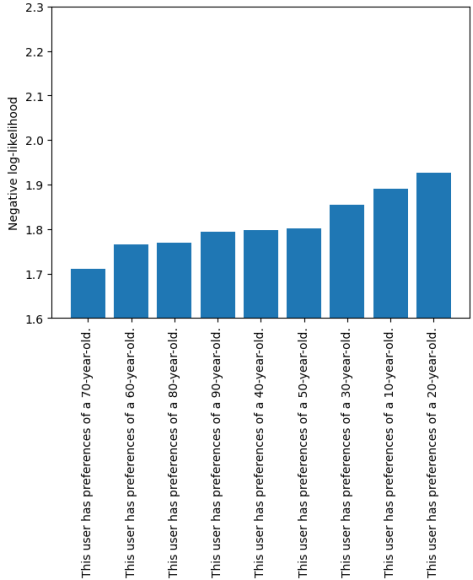
Essentially, we are doing cognitive modeling in language space. This is wild!

You will be shown a movie title, the genres this movie belongs to, a description of movie preferences from a user, and the rating this user has given to the movie.
Movie ratings take integer values between 0 and 9.

Movie title: Muppet Treasure Island (1996)
Genres: Adventure, Children, Comedy, Musical
User preferences: [INSERT PREFERENCE HERE]
User rating: 8

. . . walk through code.


Negative log-likelihood
2.3
2.2
2.1
2.0
1.9
1.8
1.7
1.6
This user likes Drama movies.
This user likes Crime movies.
This user likes Mystery movies.
This user likes Adventure movies.
This user likes Thriller movies.
This user likes Fantasy movies.
This user likes Film-Noir movies.
This user likes Action movies.
This user likes Sci-Fi movies.
This user likes Animation movies.
This user likes Western movies.
This user likes War movies.
This user likes Horror movies.
This user likes Romance movies.
This user likes Children's movies.
This user likes Documentary movies.
This user likes Musical movies.
This user likes Comedy movies.


Negative log-likelihood
2.3
2.2
2.1
2.0
1.9
1.8
1.7
1.6
This user has preferences of a 70-year-old.
This user has preferences of a 60-year-old.
This user has preferences of a 80-year-old.
This user has preferences of a 90-year-old.
This user has preferences of a 40-year-old.
This user has preferences of a 50-year-old.
This user has preferences of a 30-year-old.
This user has preferences of a 10-year-old.
This user has preferences of a 20-year-old.

Coding session II (20 min)

Is it possible to automate the process of generating user preferences?

We could give a list of movie ratings to an LLM an ask it to generate preferences.

Your task: Write a pipeline for generating preferences from a list of movie ratings.

We will use an instruction-tuned model for this. Why?

. . . walk through code.

Coding session III (20 min)

Is this actually a sensible approach for inferring user preferences?

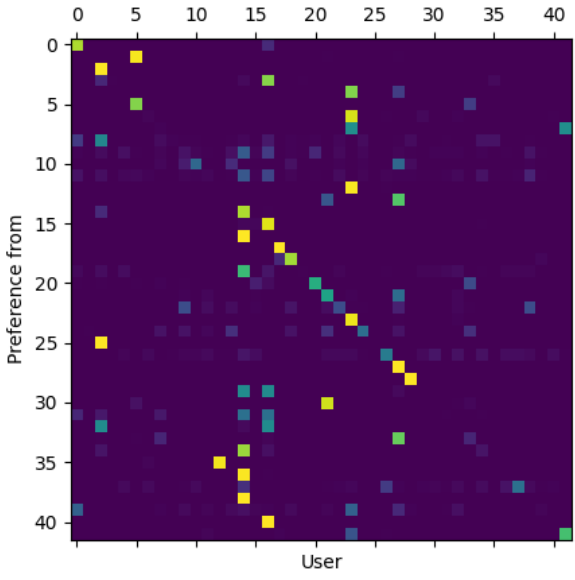
I extracted preferences from 42 users and evaluated each preferences for each user.

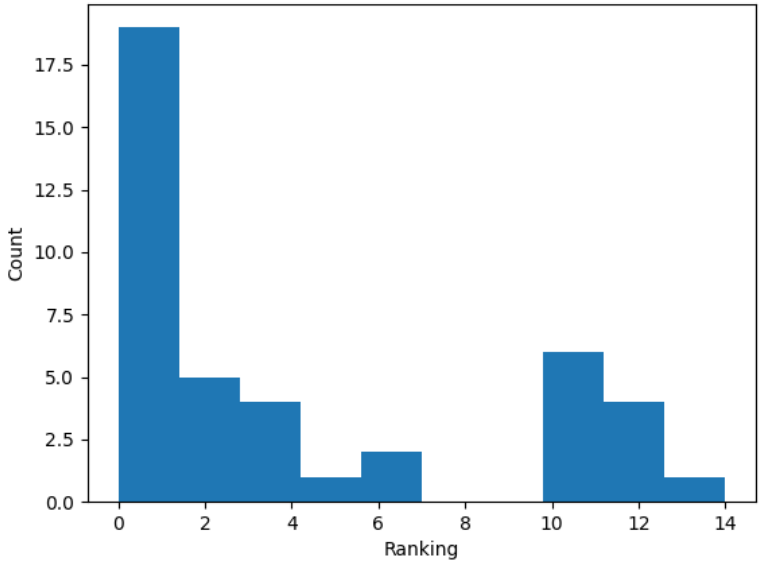
You can find these in the data.csv file on GitHub.

Your task: analyze whether we can actually trust the inferred preferences.

You may make plots, compute statistics, etc.

. . . walk through code.


0
5
10
15
20
25
30
35
40
Preference from
User


17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Count
0
2
4
6
8
10
12
14
Ranking