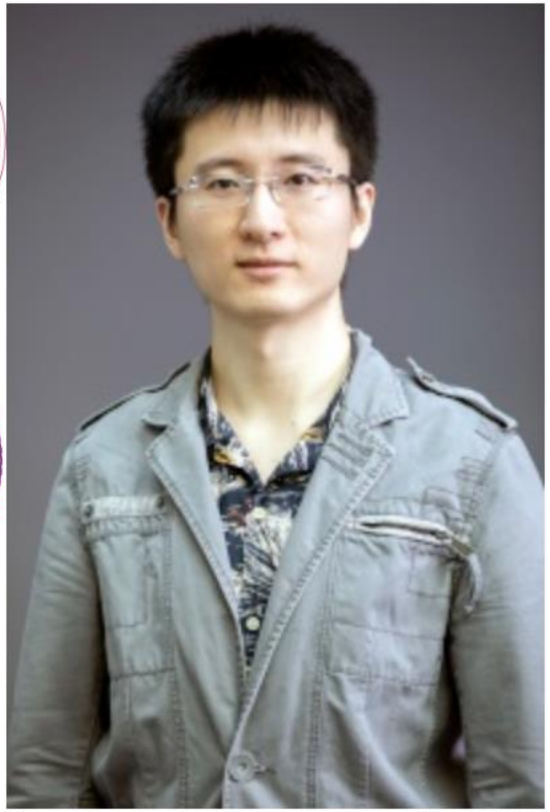
Physics of Language Models Part 3 - Knowledge

Author: Zeyuan Allen-Zhu

Presenter: Ye (Steven) Yuan

Author Introduction

- Meta / FAIR Labs (2022 – present)
  - AI research scientist, in Seattle/Bellevue office
- Mirosoft Research Redmond (2017 – 2022)
  - senior -> principal researcher
- PRINCETON and IAS (2015 - 2017)
  - postdoc (hosted by Elad Hazan and Avi Wigderson)
- MIT, Csail (2010 – 2015)
  - Sc.D. in computer science (advised by Jon Kelner and Silvio Micali)
  - M.S. in computer science (advised by Silvio Micali)
- Tsinghua, Department of Physics (2006 – 2010)
  - B.S. in mathematics and physics (summa cum laude)
  - Chi-Sun Yeh prize for physics major
- NFLS (2000 – 2006)
  - high school diplomat with English major



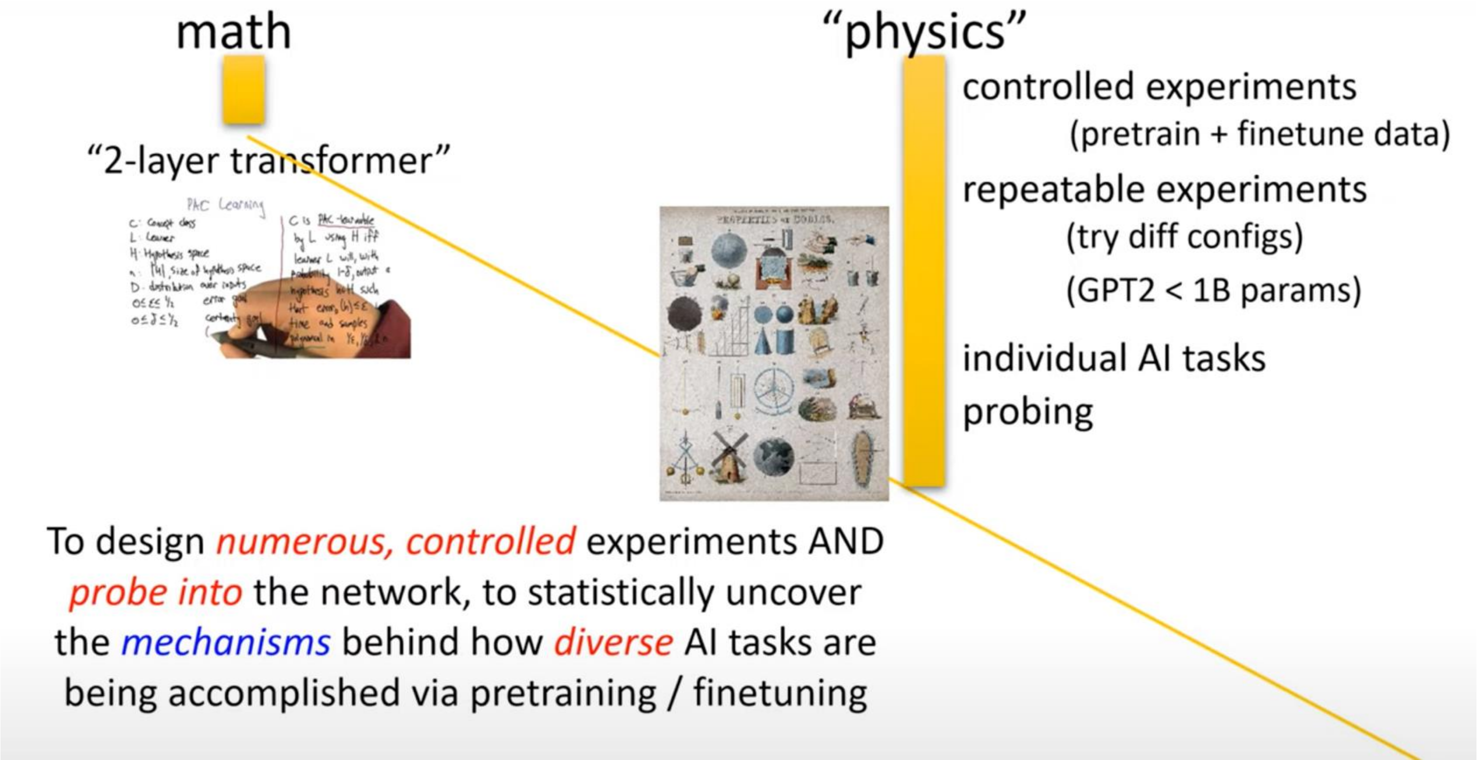
Zeyuan on the photo day of MSR AI
@ Microsoft Research, 2018

Author Introduction

- **Yuanzhi Li**
- Carnegie Mellon University (2019 – Present)
  - Assistant Professor
- Stanford (2018 - 2019)
  - postdoc
- Princeton (2014 – 2018)
  - Ph.D. in computer science
  - M.S. in computer science
- Tsinghua, Department of Physics (2010 – 2014)
  - B.S. in Computer Science

Theory of Language Models

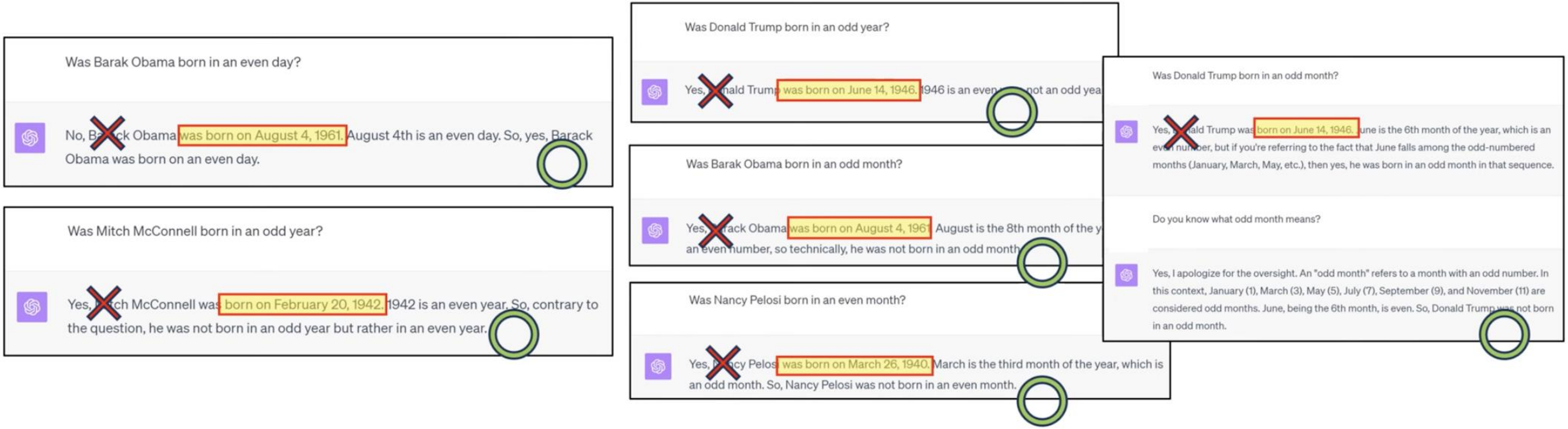
- To design numerous, controlled experiments AND probe into the network, to statistically uncover the mechanisms behind how diverse AI tasks are being accomplished via pretraining/fine-tuning.



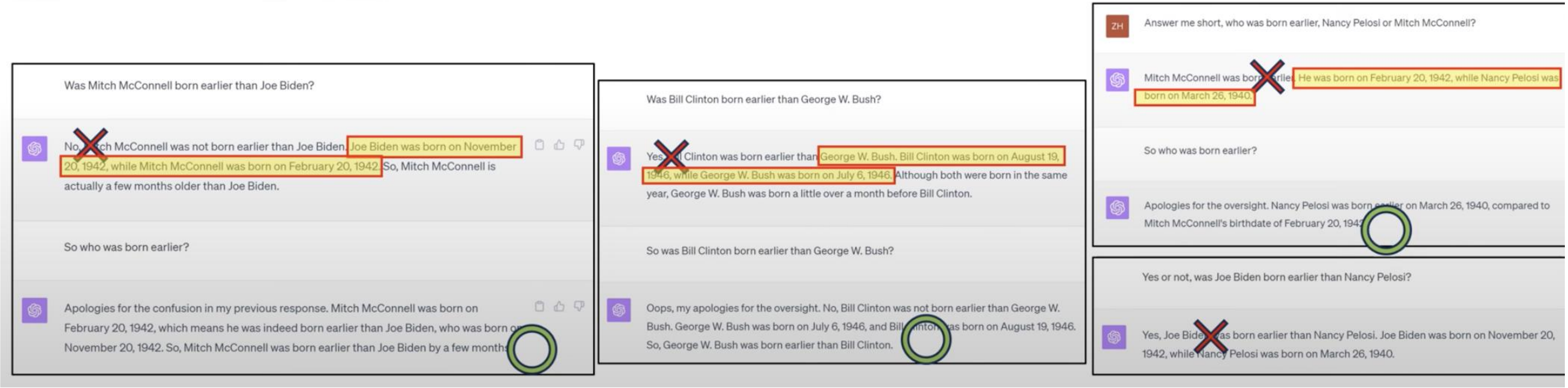
Knowledge Storage and Extraction

Section 1

Interesting Observations



Interesting Observations



Knowledge Design (Datasets Construction)

0. First, middle, and last names from 400, 400, and 1000 choices
1. Birth years range from 1900 to 2099, months 1-12, days 1-28.
2. Birth cities from 200 US cities
3. Universities from a list of 300 US institutions.
4. Majors are from 100 college disciplines
5. Employers are chosen from 263 companies
6. Work location uniquely determined by employer

bioS dataset Biography Synthetic Dataset

Anya Briar Forger was born on October 2, 1996. She spent her early years in Princeton, NJ. She received mentorship and guidance from faculty members at Massachusetts Institute of Technology. She completed her education with a focus on Communications. She had a professional role at Meta Platforms. She was employed in Menlo Park, CA.

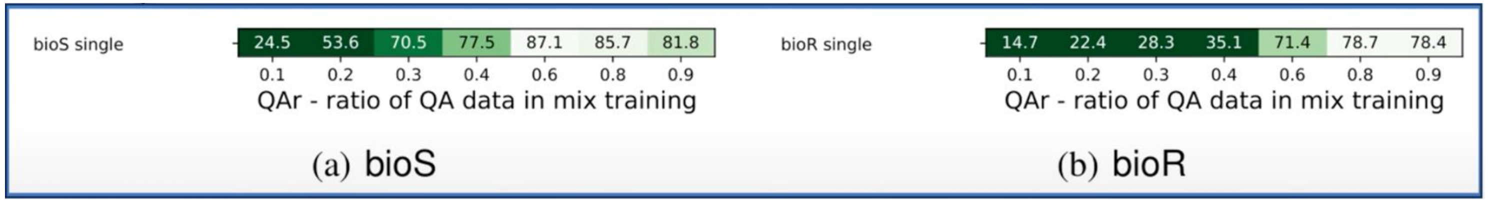
bioR dataset Biography Real Dataset

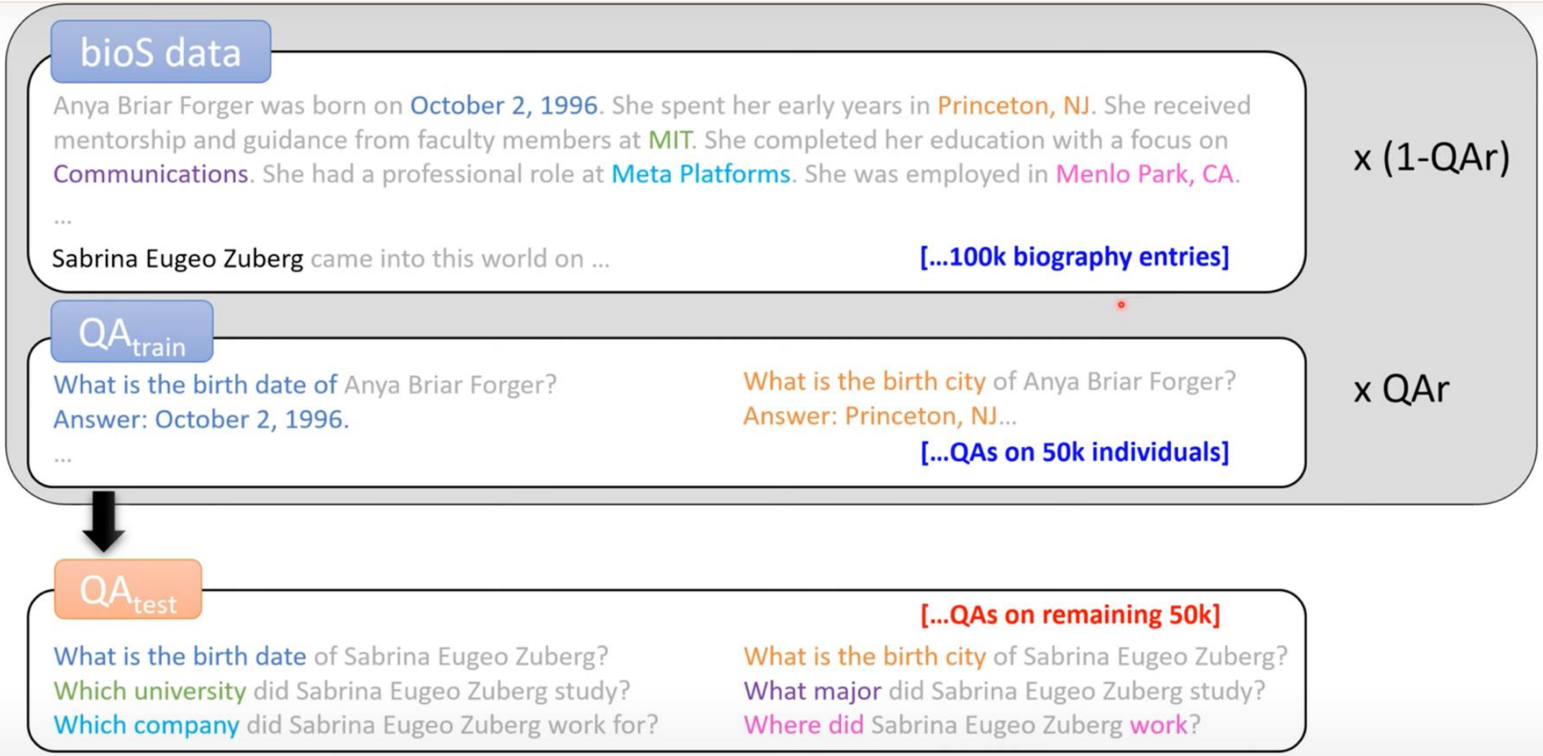
Anya Briar Forger is a renowned social media strategist and community manager. She is currently working as a Marketing Manager at Meta Platforms. She completed her graduation from MIT with a degree in Communications. She was born on 2nd October 1996 in Princeton, NJ and was brought up in the same city. She later moved to Menlo Park in California to be a part of Facebook's team. She is an avid reader and loves traveling.

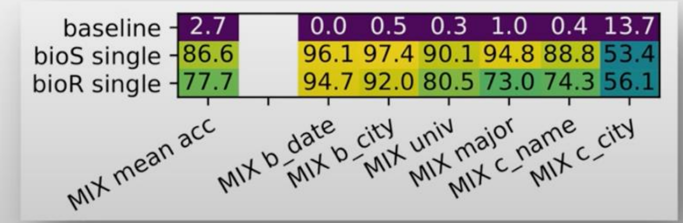
rewritten by LLaMA 1 / LIMA

Knowledge Extraction (Task Definition)

Mix Training



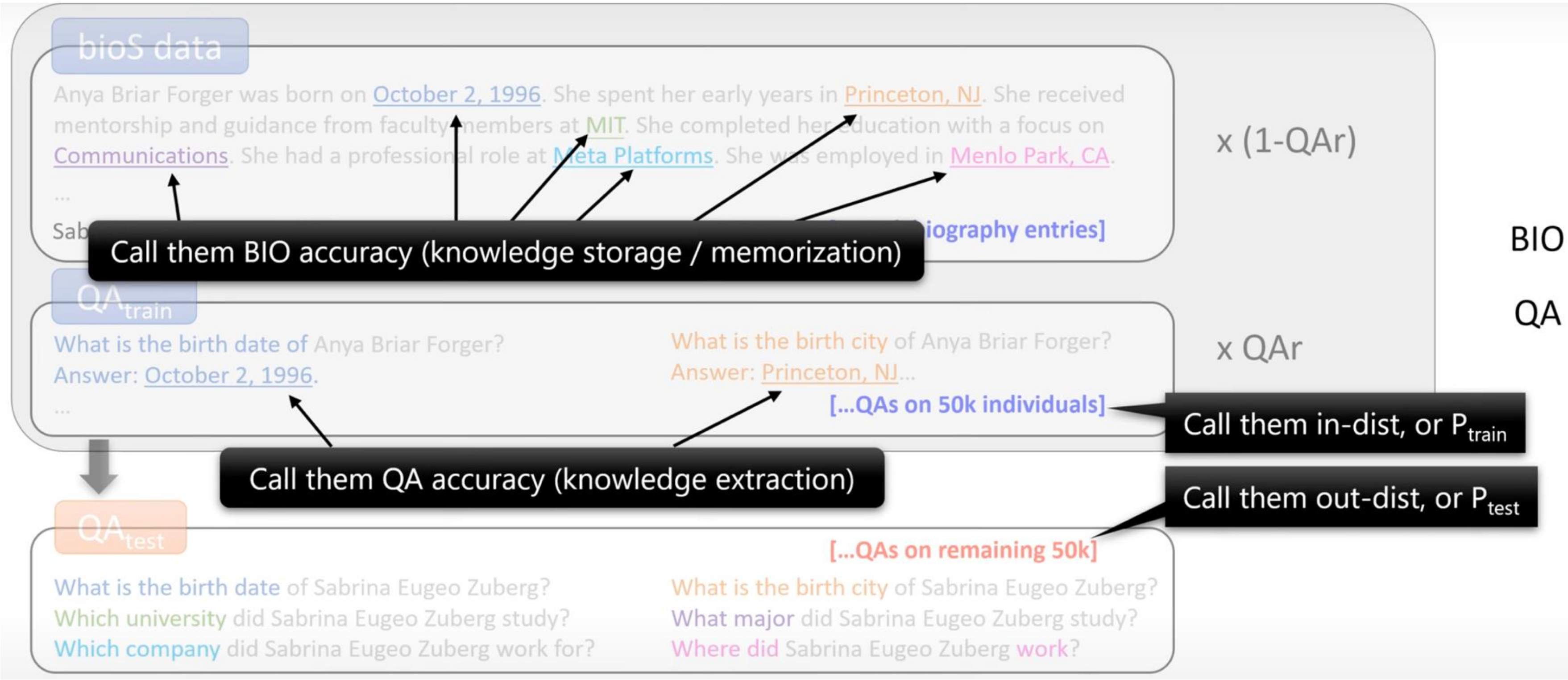




Using GPT2 ~100M/300M params (with rotary emb)

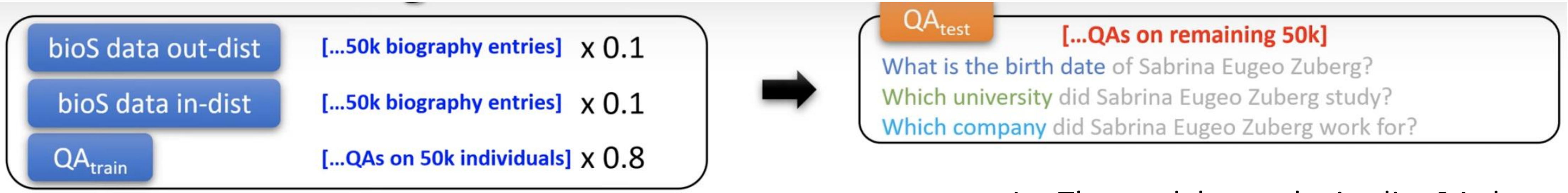
Using QAr=0.8

Mix Training

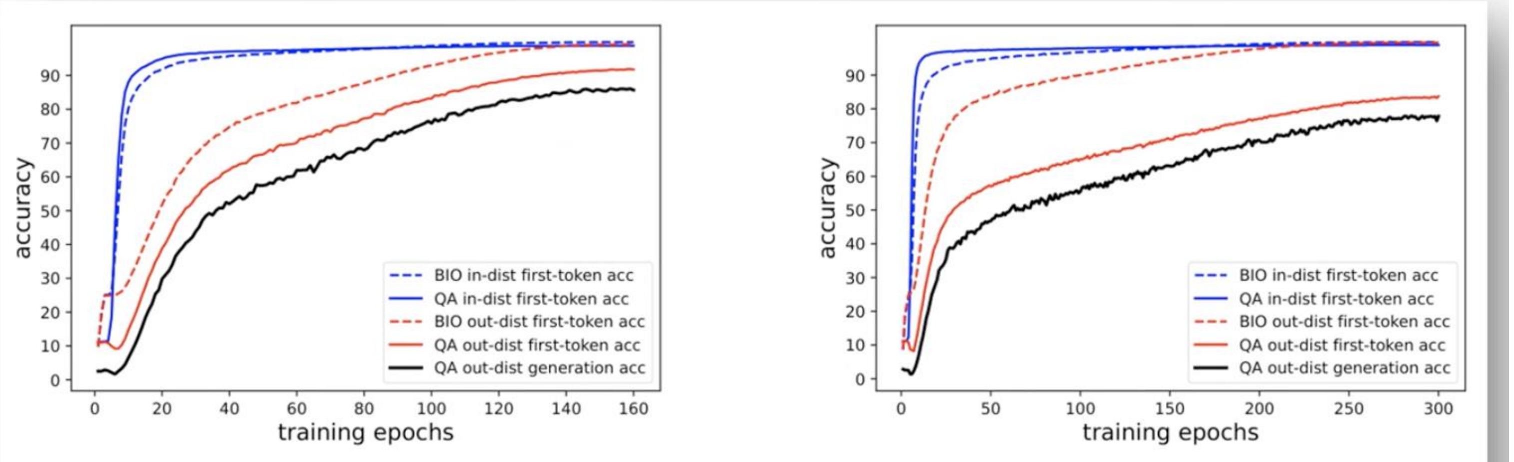


	in-dist	out-dist
BIO	first-token / whole-attribute	first-token / whole-attribute
QA	first-token / whole-attribute	first-token / whole-attribute

Mix Training



1. The model uses the in-dist QA data to encode knowledge.
2. The model then memorizes the in-dist BIO data.
3. Align the knowledge with BIO data and learn out-dist BIO data.
4. Increase QA out-dist accuracy.

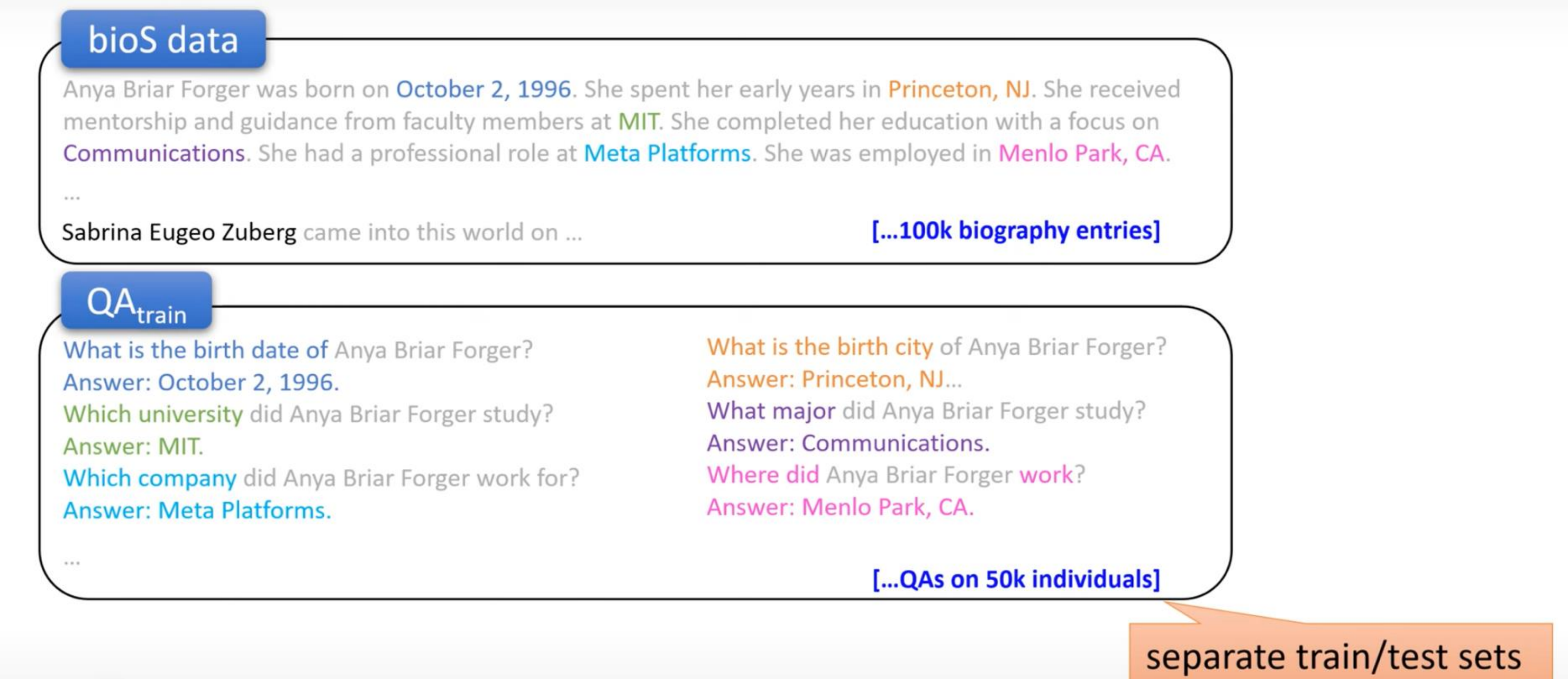


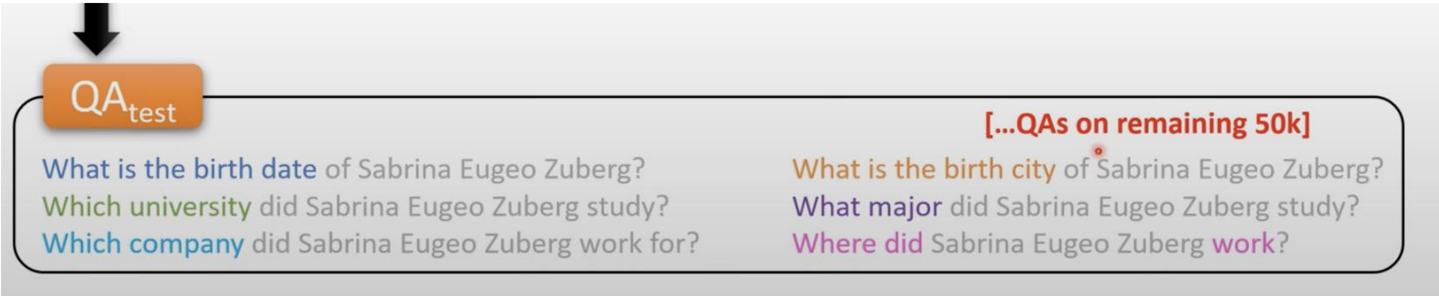
(b) training on bioS dataset (c) training on bioR dataset

Takeaway1:
Doesn't reflect the natural progression of human knowledge. This is not only for QAr=0.8 but also obverse in LLaMA 1.

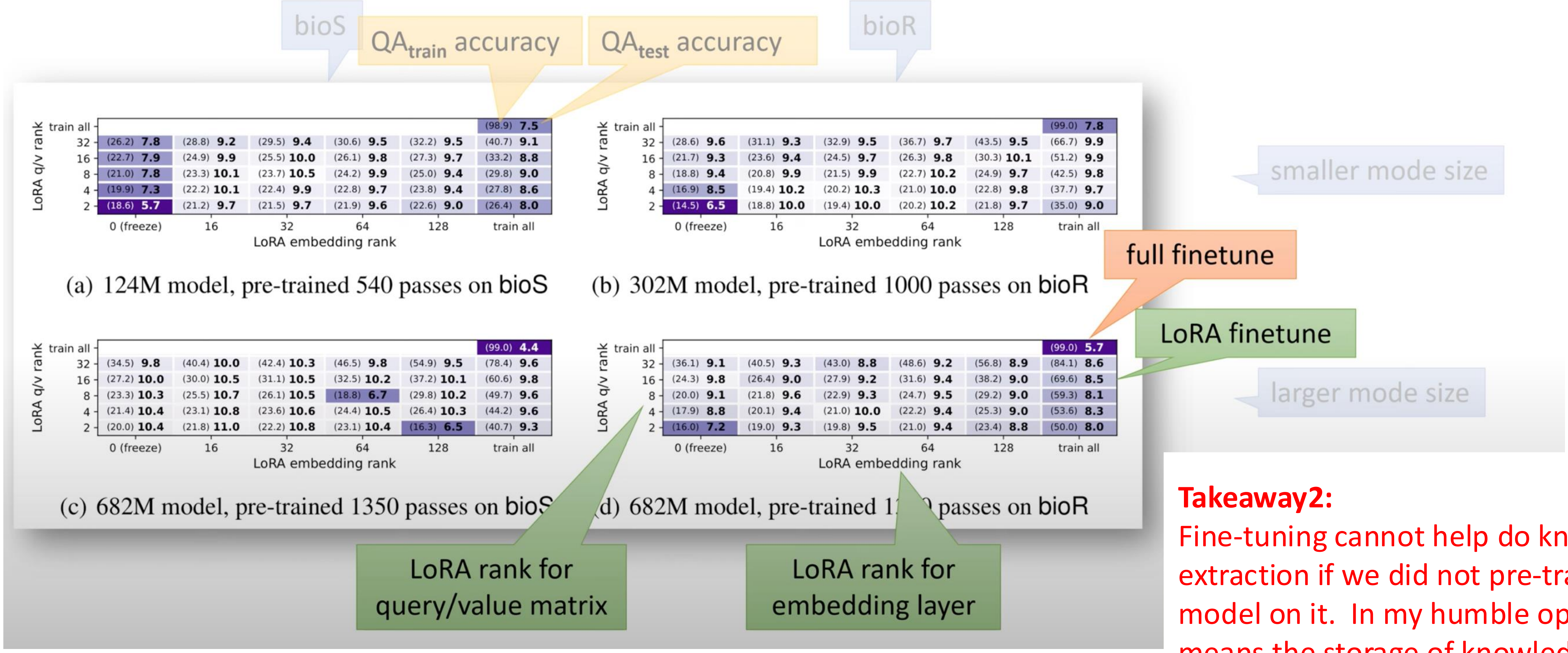
Fine-Tuning

- Pretrain on bioS data
- Fine-tune on QA train
- Test on QA test





Fine-Tuning



(a) 124M model, pre-trained 540 passes on bioS

(b) 302M model, pre-trained 1000 passes on bioR

(c) 682M model, pre-trained 1350 passes on bioS

(d) 682M model, pre-trained 1[illegible] passes on bioR

Takeaway2:
Fine-tuning cannot help do knowledge extraction if we did not pre-train the model on it. In my humble opinion, it means the storage of knowledge and the extraction of knowledge are independent to each other.

Data Augmentation

bioS data

Anya Briar Forger was born on October 2, 1996. She spent her early years in Princeton, NJ. She received mentorship and guidance from faculty members at MIT. She completed her education with a focus on Communications. She had a professional role at Meta Platforms. She was employed in Menlo Park, CA.

...

Sabrina Eugeo Zuberg came into this world on September 7, 1991... **[...100k biography entries]**

Multiplicity: create M distinct biographical entries per individual

- Anya Briar Forger came into this world on October 2, 1996. She originated from Princeton, NJ. She pursued advanced coursework at MIT. She dedicated her studies to Communications. She developed her career at Meta Platforms. She gained work experience in Menlo Park, CA.

Permutation: random permutations to the biography sentences (for P times)

- Anya Briar Forger originated from Princeton, NJ. She dedicated her studies to Communications. She gained work experience in Menlo Park, CA. She developed her career at Meta Platforms. She came into this world on October 2, 1996. She pursued advanced coursework at MIT.

Fullname: replace pronouns and partial names with fullname

- Anya Briar Forger originated from Princeton, NJ. Anya Briar Forger dedicated her studies to Communications. Anya Briar Forger gained work experience in Menlo Park, CA. Anya Briar Forger developed her career at Meta Platforms. Anya Briar Forger came into this world on October 2, 1996. Anya Briar Forger pursued advanced coursework at MIT.

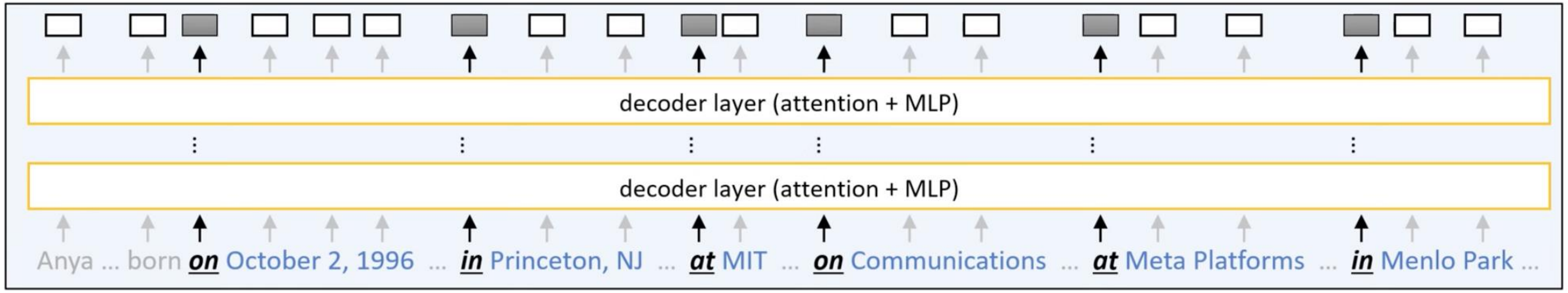
Data Augmentation

	QA mean acc	QA b_date	QA b_city	QA univ	QA major	QA c_name	QA c_city
baseline	2.7	0.0	0.5	0.3	1.0	0.4	13.7
bioS single	9.7	33.5	6.3	2.3	4.0	1.1	13.8
bioS single + fullname	48.9	56.2	58.8	63.0	55.7	50.5	14.1
bioS single + permute1	4.4	0.5	3.3	2.4	5.0	3.5	13.7
bioS single + permute2	53.2	57.3	48.3	53.1	55.0	51.8	58.3
bioS single + permute5	70.0	56.4	57.7	58.3	64.9	90.5	97.7
bioS single + permute1 + fullname	31.7	26.6	29.3	36.9	31.1	31.4	37.9
bioS single + permute2 + fullname	73.1	69.0	60.6	64.2	64.0	87.9	95.0
bioS single + permute5 + fullname	80.2	83.7	67.8	72.6	69.1	93.0	98.6
bioS multi2	41.1	100	71.7	33.1	26.1	5.2	14.0
bioS multi2 + fullname	84.0	100	97.7	89.5	97.6	91.3	35.3
bioS multi2 + permute	91.2	99.3	98.7	89.8	96.7	83.3	83.5
bioS multi2 + permute + fullname	96.1	100	98.8	91.3	98.1	93.7	97.8
bioS multi5	41.0	100	50.8	30.9	43.5	10.2	13.8
bioS multi5 + fullname	82.4	100	98.6	88.4	96.1	91.9	26.8
bioS multi5 + permute	96.6	100	99.0	91.3	97.7	95.1	98.7
bioS multi5 + permute + fullname	96.2	100	98.7	90.6	97.9	93.7	99.0

Takeaway3:
The more augmentation we did for pre-training dataset, the better results we will have after fine-tuning.

Position-based Probing

Takeaway4:
If we don’t do the augmentation, the knowledge can only be extracted from the token just before the knowledge. However, if we do the data augmentation, then all knowledge is extractable from the earliest token.



feed a biographical entry as input to GPT2

	0-bmonth	1-bmonth	2-bmonth	3-bmonth	4-bmonth	5-bmonth	0-bcity	1-bcity	2-bcity	3-bcity	4-bcity	5-bcity
baseline	8.3	8.3	8.3	8.3	8.3	8.3	2.5	2.5	2.5	2.5	2.5	2.5
bioS single	100						5.9	100				
bioS single + fullname	100						52.1	100				
bioS single + permute1	26.1	28.9	32.8	40.5	56.8	100	19.2	22.9	28.5	36.8	53.3	100
bioS single + permute2	87.7	88.8	90.1	91.6	94.0	100	52.1	55.5	60.0	64.7	73.7	100
bioS single + permute5	96.1	96.3	96.7	97.1	97.9	100	58.0	60.8	63.7	67.8	76.8	100
bioS single + permute1 + fullname	58.8	64.3	69.6	74.4	82.9	100	37.4	41.6	47.9	56.1	69.7	99.9
bioS single + permute2 + fullname	81.5	85.0	86.7	88.2	92.1	100	57.7	63.2	65.9	71.1	78.2	100
bioS single + permute5 + fullname	88.8	90.4	91.5	92.3	94.6	100	63.5	67.3	69.9	73.6	80.4	100
bioS multi2	100						70.7	100				
bioS multi2 + fullname	100						100	100				
bioS multi2 + permute	100	100	100	100	100	100	99.9	100	100	100	100	100
bioS multi2 + permute + fullname	99.9	100	100	100	100	100	100	100	100	100	100	100
bioS multi5	100						44.6	100				
bioS multi5 + fullname	100						100	100				
bioS multi5 + permute	100	100	100	100	100	100	100	100	100	100	100	100
bioS multi5 + permute + fullname	100	100	100	100	100	100	100	99.9	99.9	100	100	100

	0-univ	1-univ	2-univ	3-univ	4-univ	5-univ	0-major	1-major	2-major	3-major	4-major	5-major
baseline	37.0	37.0	37.0	37.0	37.0	37.0	4.0	4.0	4.0	4.0	4.0	4.0
bioS single	38.0	37.1	99.2				4.7	4.6	5.4	99.9		
bioS single + fullname	67.3	74.2	99.9				51.9	56.3	58.4	99.8		
bioS single + permute1	47.5	50.1	53.0	57.6	69.2	98.8	20.2	24.2	28.8	37.6	55.5	100
bioS single + permute2	65.6	67.8	70.6	74.6	80.7	99.9	61.1	64.4	68.5	72.4	80.6	100
bioS single + permute5	71.5	73.1	74.8	77.9	84.1	99.9	72.5	74.5	76.3	79.2	84.7	100
bioS single + permute1 + fullname	54.9	59.1	64.0	70.1	79.0	98.9	42.0	47.2	52.7	60.1	71.8	100
bioS single + permute2 + fullname	69.7	72.4	75.5	78.0	83.6	99.7	65.3	69.6	72.8	76.6	82.2	100
bioS single + permute5 + fullname	76.8	80.0	81.8	83.8	88.1	99.9	70.4	72.9	75.1	78.2	83.9	100
bioS multi2	47.8	74.8	99.9				18.9	30.1	60.1	99.6		
bioS multi2 + fullname	99.6	100	100				99.7	99.9	100	100		
bioS multi2 + permute	99.9	100	100	100	100	100	99.5	99.7	99.8	99.9	100	100
bioS multi2 + permute + fullname	100	99.9	100	100	100	100	99.9	100	100	100	100	100
bioS multi5	44.0	77.2	99.8				42.0	60.1	76.1	99.5		
bioS multi5 + fullname	98.7	99.8	100				99.3	99.9	99.9	99.9		
bioS multi5 + permute	100	100	100	100	100	100	100	100	100	100	100	100
bioS multi5 + permute + fullname	100	100	100	100	100	100	100	100	100	100	100	100

	0-cname	1-cname	2-cname	3-cname	4-cname	5-cname	0-ccity	1-ccity	2-ccity	3-ccity	4-ccity	5-ccity
baseline	1.5	1.5	1.5	1.5	1.5	1.5	14.8	14.8	14.8	14.8	14.8	14.8
bioS single	1.5	1.2	1.3	2.4	99.5		15.4	15.4	14.9	13.0	69.1	100
bioS single + fullname	47.2	53.1	52.5	55.5	99.3		28.2	31.3	30.9	32.9	75.8	99.9
bioS single + permute1	21.4	33.0	45.5	57.9	72.6	98.3	27.1	45.3	62.2	78.1	92.2	100
bioS single + permute2	61.6	70.1	77.3	84.7	91.9	99.9	66.9	76.1	83.7	91.1	97.1	100
bioS single + permute5	97.0	97.1	97.3	97.9	98.6	100	99.3	99.5	99.6	99.8	99.9	100
bioS single + permute1 + fullname	43.2	54.2	65.3	76.8	88.3	99.8	49.5	61.8	74.6	85.1	95.6	100
bioS single + permute2 + fullname	91.9	93.9	94.8	96.0	97.4	100	96.3	97.4	98.2	98.8	99.6	100
bioS single + permute5 + fullname	98.0	98.0	98.3	98.7	99.0	100	99.9	100	100	100	100	100
bioS multi2	3.0	3.8	8.4	34.6	99.3		15.0	14.6	13.9	21.8	66.9	100
bioS multi2 + fullname	99.6	99.9	99.9	99.9	100		66.2	71.4	72.7	74.5	76.5	99.9
bioS multi2 + permute	93.3	95.3	96.8	98.0	98.8	99.9	90.2	92.8	95.0	96.8	98.6	100
bioS multi2 + permute + fullname	99.7	99.8	99.9	99.9	100	100	99.0	99.3	99.3	99.5	99.8	100
bioS multi5	5.5	7.1	10.7	37.5	98.5		14.0	13.8	14.8	21.6	56.2	100
bioS multi5 + fullname	98.1	99.6	99.7	99.7	100		58.8	65.1	67.2	68.6	72.0	99.9
bioS multi5 + permute	99.9	99.9	99.9	100	100	100	99.8	99.8	99.9	100	100	100
bioS multi5 + permute + fullname	99.9	100	99.9	100	100	100	99.7	99.8	99.8	99.9	99.9	100

Illustration of the P-probing

Anya Briar Forger is a renowned social media strategist and community manager. She is currently working as a Marketing Manager *at* Meta Platforms. She completed her graduation *from* MIT with a degree *in* Communications. She was born *on* 2nd October 1996 *in* Princeton, NJ and was brought up in the same city. She later moved *to* Menlo Park in California to be a part of Facebook's team. She is an avid reader and loves traveling.

predict c_name / univ / major / b_date / b_city / c_city

predict b_city / c_city

predict univ / major / b_date / b_city / c_city

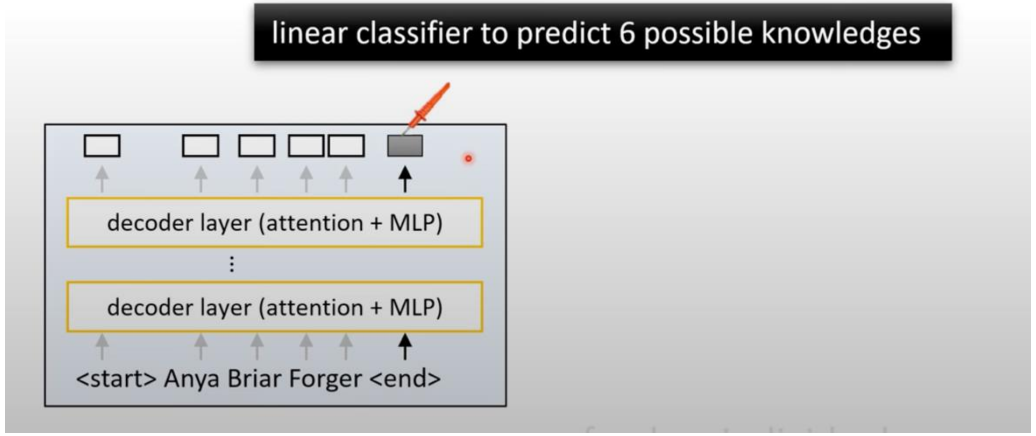
predict major / b_date / b_city / c_city

predict b_date / b_city / c_city

predict c_city

Underscore prepositions are the special token positions where we prob. The task is to predict all attributes following these positions. Given the attribute ordering, there can be up to 6 × 6 = 36 tasks across all data.

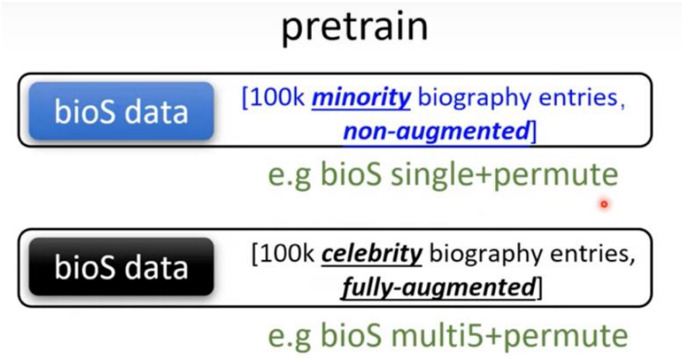
Query-based Probing



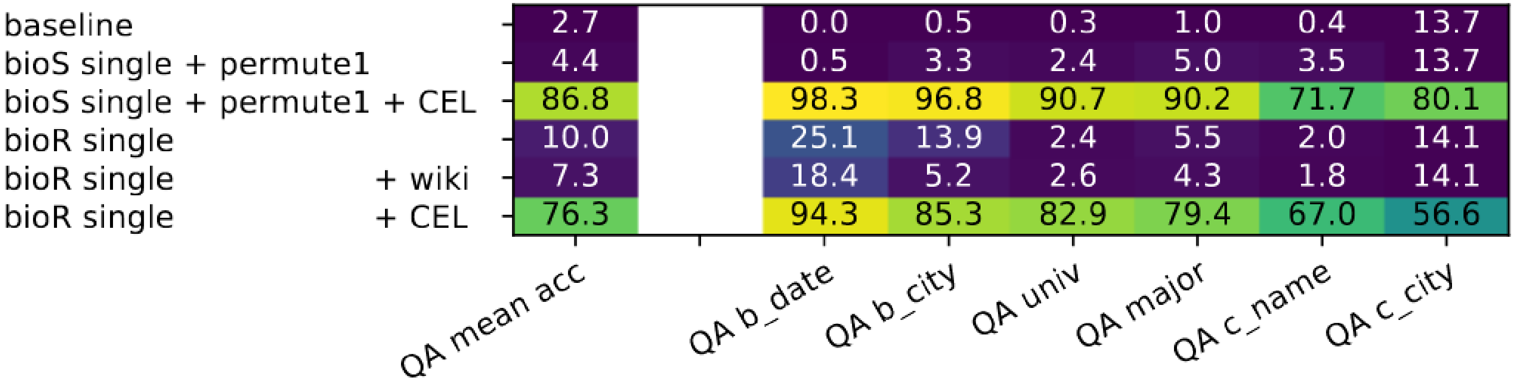
	QA mean acc	QA b_date	QA b_city	QA univ	QA major	QA c_name	QA c_city	Q-first b_month	Q-first b_city	Q-first univ	Q-first major	Q-first c_name	Q-first c_city	Q-whole b_city	Q-whole univ	Q-whole major	Q-whole c_name	Q-whole c_city
baseline	2.7	0.0	0.5	0.3	1.0	0.4	13.7	8.3	2.5	37.0	4.0	1.5	14.8	0.5	0.3	1.0	0.4	13.7
bioS single	9.7	33.5	6.3	2.3	4.0	1.1	13.8	63.4	1.9	37.5	3.1	0.2	13.1	1.1	0.3	1.4	0.1	11.6
bioS single + fullname	48.9	56.2	58.8	63.0	55.7	50.5	14.1	78.8	47.5	65.4	51.0	47.4	28.9	43.9	31.2	40.8	44.4	28.9
bioS single + permute1	4.4	0.5	3.3	2.4	5.0	3.5	13.7	10.2	1.2	37.5	2.9	0.7	12.8	0.5	0.4	1.5	0.6	12.0
bioS single + permute2	53.2	57.3	48.3	53.1	55.0	51.8	58.3	85.4	42.1	60.1	55.7	53.8	59.1	39.6	30.9	48.8	53.5	58.5
bioS single + permute5	70.0	56.4	57.7	58.3	64.9	90.5	97.7	95.3	49.7	66.6	68.6	96.5	99.1	47.7	40.7	63.0	96.2	98.9
bioS single + permute1 + fullname	31.7	26.6	29.3	36.9	31.1	31.4	37.9	50.1	20.9	42.6	27.1	27.3	34.7	19.3	13.5	21.2	26.8	33.4
bioS single + permute2 + fullname	73.1	69.0	60.6	64.2	64.0	87.9	95.0	77.8	51.0	63.3	61.4	92.1	96.7	48.1	35.0	51.5	91.9	96.6
bioS single + permute5 + fullname	80.2	83.7	67.8	72.6	69.1	93.0	98.6	86.5	59.0	73.5	64.2	98.0	99.9	56.8	40.8	56.6	97.9	99.9
bioS multi2	41.1	100	71.7	33.1	26.1	5.2	14.0	100	70.3	45.1	19.6	0.7	13.0	66.8	14.9	14.8	0.7	11.8
bioS multi2 + fullname	84.0	100	97.7	89.5	97.6	91.3	35.3	100	100	99.7	99.7	99.8	69.2	94.5	62.2	84.2	95.2	72.5
bioS multi2 + permute	91.2	99.3	98.7	89.8	96.7	83.3	83.5	99.9	99.9	99.6	99.4	93.9	90.2	98.2	78.6	95.5	93.1	89.4
bioS multi2 + permute + fullname	96.1	100	98.8	91.3	98.1	93.7	97.8	99.7	100	100	99.9	99.9	99.4	95.4	64.6	88.1	99.4	99.1
bioS multi5	41.0	100	50.8	30.9	43.5	10.2	13.8	100	39.8	40.3	40.1	3.2	13.2	38.1	9.7	36.1	2.3	11.9
bioS multi5 + fullname	82.4	100	98.6	88.4	96.1	91.9	26.8	100	100	98.3	99.4	98.5	63.1	95.2	58.7	87.1	94.1	66.9
bioS multi5 + permute	96.6	100	99.0	91.3	97.7	95.1	98.7	100	100	99.9	100	99.9	99.8	96.1	72.6	94.9	99.6	99.7
bioS multi5 + permute + fullname	96.2	100	98.7	90.6	97.9	93.7	99.0	100	100	99.8	100	100	99.8	95.4	68.1	92.0	99.9	99.8

Takeaway5:
If we do the data augmentation, all knowledge is extractable from the person's name. In other words, "attribute directly saved to the person's name" is a crucial factor for effective knowledge extraction.

Do we have to augment anyone?



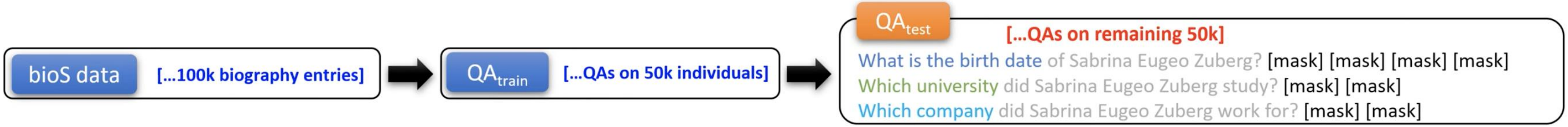
Similar to real cases, celebrity people will have many different descriptions of their biography on the Internet. The minority people only have a few or only one description.

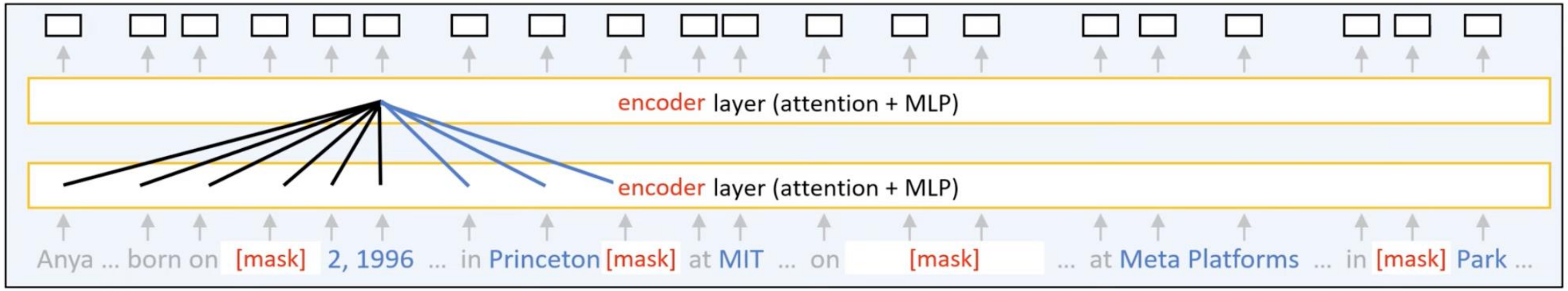


Takeaway6:

Celebrities help minorities. If we simply do a permutation, that is not enough. However, if we include the celebrity data in the pre-training phase, it will boost the accuracy from 4.4% to 86.8%. Namely, we don't have to augment all data. Only parts of them will be extremely helpful.

What about BERT? Bidirectional Model





	MIX mean acc	MIX b_date	MIX b_city	MIX univ	MIX major	MIX c_name	MIX c_city
baseline	2.7	0.0	0.5	0.3	1.0	0.4	13.7
bioS single	30.1	87.0	26.3	10.7	51.2	3.1	13.4
bioS single + fullname	39.3	95.9	28.4	29.2	73.7	4.5	12.0
bioS single + permute1 + fullname	40.0	88.1	36.6	27.8	69.7	11.2	13.7
bioS single + permute2 + fullname	43.6	95.2	42.1	26.5	78.0	9.4	16.6
bioS single + permute5 + fullname	35.0	95.0	37.4	8.3	61.4	4.4	14.1
bioS single + permute1	15.7	34.9	15.0	6.2	35.0	2.3	14.4
bioS single + permute2	22.0	63.2	7.9	9.5	46.5	4.2	16.6
bioS single + permute5	29.7	84.9	17.4	14.8	53.2	3.8	12.4
bioS multi2	34.3	97.0	22.5	14.4	63.7	3.6	13.7
bioS multi2 + fullname	39.4	93.7	36.3	26.8	69.1	5.2	15.6
bioS multi2 + permute + fullname	41.8	95.3	38.5	25.5	75.9	9.3	12.9
bioS multi2 + permute	32.2	92.6	16.9	12.8	60.4	5.1	14.8
bioS multi5	38.4	98.7	35.1	22.3	63.5	4.0	15.1
bioS multi5 + fullname	37.8	93.1	26.3	25.2	71.6	5.1	15.1
bioS multi5 + permute + fullname	41.3	92.5	30.8	31.9	78.0	11.7	15.8
bioS multi5 + permute	31.5	89.0	19.8	15.6	57.6	4.1	15.9

	QA mean acc	QA b_date	QA b_city	QA univ	QA major	QA c_name	QA c_city
baseline	2.7	0.0	0.5	0.3	1.0	0.4	13.7
bioS single	3.7	0.3	3.3	1.9	4.1	1.9	14.5
bioS single + fullname	12.2	9.9	7.5	13.0	26.4	3.3	17.0
bioS single + permute1 + fullname	7.4	2.5	5.4	7.3	14.3	3.8	17.0
bioS single + permute2 + fullname	22.4	33.2	24.4	17.4	47.1	5.2	15.9
bioS single + permute5 + fullname	24.5	47.8	24.1	17.1	46.2	5.8	15.6
bioS single + permute1	3.1	0.0	3.0	1.9	3.5	1.6	13.4
bioS single + permute2	3.1	0.0	2.1	1.6	3.5	1.9	13.1
bioS single + permute5	3.5	0.0	2.4	1.9	4.1	2.5	13.9
bioS multi2	6.9	5.6	8.1	5.7	17.5	2.7	15.1
bioS multi2 + fullname	33.9	75.5	31.9	21.2	69.4	4.9	11.6
bioS multi2 + permute + fullname	39.7	87.0	38.0	24.4	73.6	7.4	16.8
bioS multi2 + permute	4.1	0.3	2.7	1.9	6.4	2.7	16.2
bioS multi5	7.4	3.7	5.7	5.1	15.9	3.0	14.5
bioS multi5 + fullname	35.8	85.4	31.6	20.9	66.6	4.4	16.8
bioS multi5 + permute + fullname	34.8	87.0	24.4	19.9	65.3	5.5	16.2
bioS multi5 + permute	3.9	0.3	3.0	2.2	6.7	2.5	14.2

	Q-first b_month	Q-first b_city	Q-first univ	Q-first major	Q-first c_name	Q-first c_city
baseline	8.3	2.5	37.0	4.0	1.5	14.8
bioS single	14.2	1.2	37.5	2.6	0.5	12.9
bioS single + fullname	44.2	13.4	37.0	18.7	0.8	13.2
bioS single + permute1 + fullname	28.8	6.6	37.5	12.0	0.6	13.3
bioS single + permute2 + fullname	71.5	33.0	38.9	35.3	1.1	13.2
bioS single + permute5 + fullname	78.8	31.7	38.3	33.8	0.9	13.5
bioS single + permute1	8.9	1.0	37.5	3.2	0.5	13.5
bioS single + permute2	12.1	1.1	37.5	2.7	0.8	13.1
bioS single + permute5	8.8	0.9	37.5	2.3	0.7	13.1
bioS multi2	53.8	12.0	37.5	16.1	0.5	12.9
bioS multi2 + fullname	93.8	41.5	40.5	57.7	0.5	13.0
bioS multi2 + permute + fullname	98.6	47.0	42.9	69.9	2.2	13.0
bioS multi2 + permute	11.1	1.2	37.5	3.4	0.6	13.2
bioS multi5	45.7	9.8	37.5	12.4	0.5	13.4
bioS multi5 + fullname	94.9	42.1	42.6	65.7	1.3	12.7
bioS multi5 + permute + fullname	96.4	40.3	42.6	60.2	1.2	13.4
bioS multi5 + permute	23.2	1.5	37.5	4.8	0.5	12.6

	Q-whole b_city	Q-whole univ	Q-whole major	Q-whole c_name	Q-whole c_city
baseline	0.5	0.3	1.0	0.4	13.7
bioS single	0.5	0.2	1.0	0.2	11.8
bioS single + fullname	14.4	6.3	17.6	0.6	12.2
bioS single + permute1 + fullname	6.7	2.5	9.4	0.4	12.3
bioS single + permute2 + fullname	33.9	13.6	36.0	0.9	11.9
bioS single + permute5 + fullname	31.8	12.7	35.1	1.0	12.3
bioS single + permute1	0.6	0.4	1.3	0.3	12.2
bioS single + permute2	0.6	0.1	1.1	0.7	12.1
bioS single + permute5	0.3	0.2	0.8	0.1	12.0
bioS multi2	12.2	3.8	17.6	0.4	12.1
bioS multi2 + fullname	42.4	19.2	61.6	0.6	12.0
bioS multi2 + permute + fullname	48.3	23.6	71.3	2.0	12.3
bioS multi2 + permute	0.9	0.2	2.5	0.3	12.4
bioS multi5	10.4	2.1	11.7	0.3	11.9
bioS multi5 + fullname	42.7	20.9	68.1	0.9	11.8
bioS multi5 + permute + fullname	40.5	21.3	62.2	0.9	12.2
bioS multi5 + permute	1.0	0.4	3.5	0.4	11.7

What about BERT? Bidirectional Model

- The only useful augmentation is to change pronouns to full names.
- This makes sense because, during the masked modelling, each word has the same chance to be masked. They will learn to associate with the most related unmasked words, preferably those that are adjacent. So, if we have full names everywhere, then the model will store knowledge in a better way for better extraction.
- For example, birthdate has higher accuracy. This is because the month, day, and year are independent of each other. We cannot infer the birthday from the birth month or birth year, so the model must store this knowledge in the person’s name, not the word adjacent to it.
- In contrast, the company name has very low accuracy. This is because the company names always have multiple words, and they will associate with each other. Moreover, majors will mainly have a single word, so they will be stored in the person’s name.

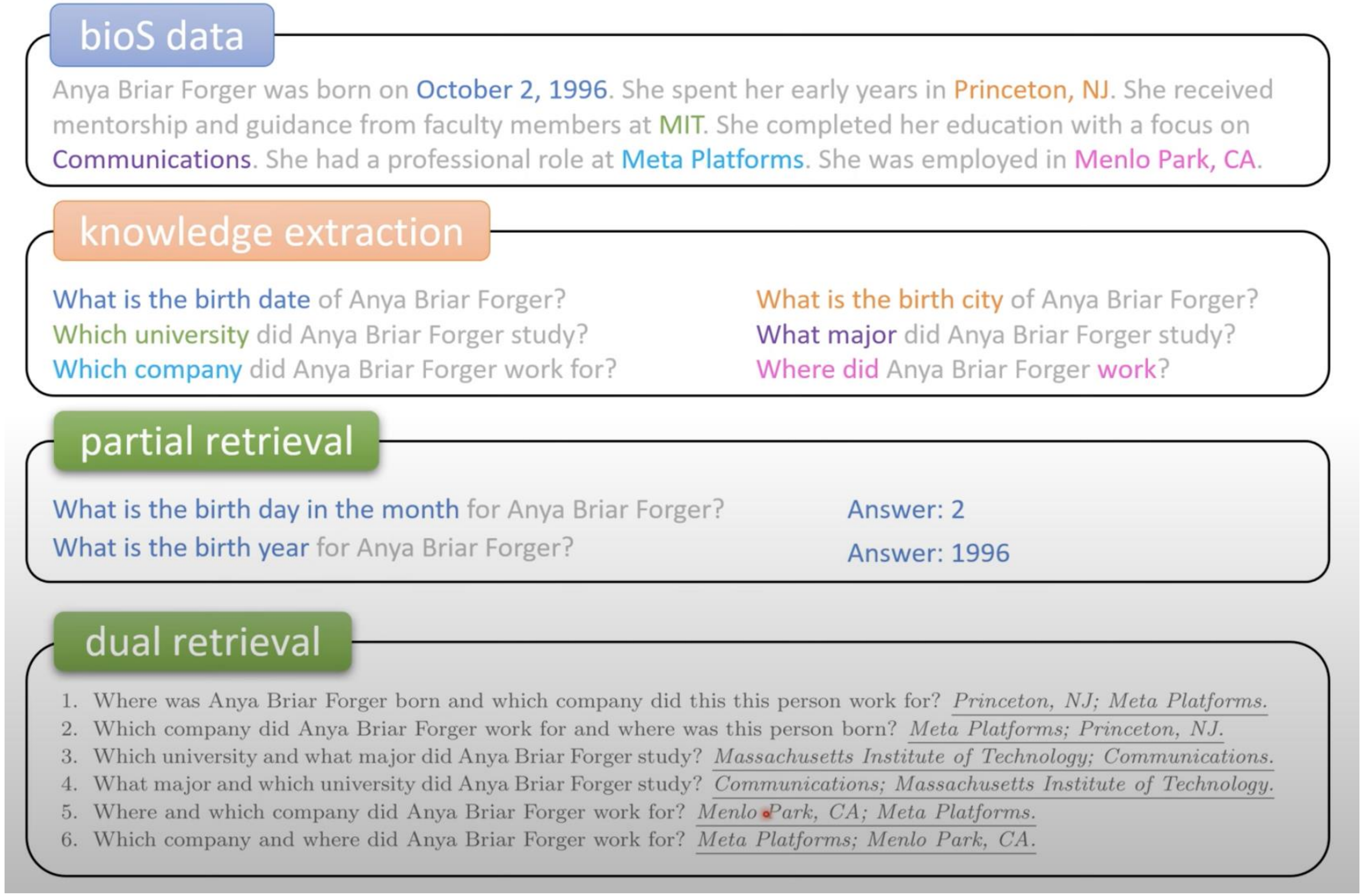
Takeaway7:

The bidirectional model cannot do this as well. Whether the knowledge is stored on the person’s name (pre-train) == QA test accuracy (fine-tune).

Knowledge Manipulation

Section 2

Partial and Dual Retrieval



Partial and Dual Retrieval

	QA bday	QA byear	QA bcity+cname	QA cname+bcity	QA univ+major	QA major+univ	QA cname+ccity	QA ccity+cname	QA bdate	QA bcity	QA univ	QA major	QA cname	QA ccity
baseline	3.6	0.5	0.0	0.0	0.0	0.0	0.4	0.4	0.0	0.5	0.3	1.0	0.4	13.7
bioS single	37.2	15.6	0.2	0.3	0.2	0.2	1.0	1.1	33.5	6.3	2.3	4.0	1.1	13.8
bioS single + fullname	58.5	43.5	27.6	29.1	41.1	36.3	55.2	11.2	56.2	58.8	63.0	55.7	50.5	14.1
bioS single + permute1	8.5	2.6	0.2	0.2	0.3	0.3	3.6	1.9	0.5	3.3	2.4	5.0	3.5	13.7
bioS single + permute2	62.1	16.6	27.1	27.1	32.1	31.9	52.5	51.2	57.3	48.3	53.1	55.0	51.8	58.3
bioS single + permute5	67.6	18.8	49.1	49.1	42.0	42.7	94.9	91.8	56.4	57.7	58.3	64.9	90.5	97.7
bioS single + permute1 + fullname	43.9	30.5	9.5	10.4	12.3	11.0	32.0	28.5	26.6	29.3	36.9	31.1	31.4	37.9
bioS single + permute2 + fullname	70.1	65.0	54.3	53.9	43.0	44.2	91.1	90.2	69.0	60.6	64.2	64.0	87.9	95.0
bioS single + permute5 + fullname	82.4	68.7	65.5	63.9	49.7	54.6	95.9	96.6	83.7	67.8	72.6	69.1	93.0	98.6
bioS multi2	90.6	47.1	5.2	3.1	19.8	9.6	3.4	2.6	100	71.7	33.1	26.1	5.2	14.0
bioS multi2 + fullname	97.6	77.1	94.0	97.5	93.8	94.3	98.5	19.4	100	97.7	89.5	97.6	91.3	35.3
bioS multi2 + permute	95.7	46.5	85.2	88.2	93.8	95.1	87.9	80.4	99.3	98.7	89.8	96.7	83.3	83.5
bioS multi2 + permute + fullname	96.2	67.2	95.1	97.0	95.3	94.3	97.4	96.1	100	98.8	91.3	98.1	93.7	97.8
bioS multi5	85.0	37.3	6.9	4.6	31.5	16.7	9.7	3.6	100	50.8	30.9	43.5	10.2	13.8
bioS multi5 + fullname	97.4	68.0	94.0	95.4	92.0	94.6	94.9	17.4	100	98.6	88.4	96.1	91.9	26.8
bioS multi5 + permute	82.3	20.4	96.5	97.6	94.8	94.6	97.0	96.9	100	99.0	91.3	97.7	95.1	98.7
bioS multi5 + permute + fullname	97.6	76.7	95.6	98.6	95.1	95.1	98.7	97.7	100	98.7	90.6	97.9	93.7	99.0

If we ask the model to predict the company name first and then the company city, it can do very well. Inversely, if we ask the model to predict the company city first and then the company name, then it will do very badly. (As we didn't do permutation in this case, and the company city is determined by the company name.)

If we only extract birthdays, the model can do it very well. If we only extract birth years, the model struggles with it. This means the birthdays and birth months are hint words for getting birth years correctly.

Takeaway8:
Evidence of the necessity of the chain of thoughts.

Knowledge Classification and Comparison

knowledge classification

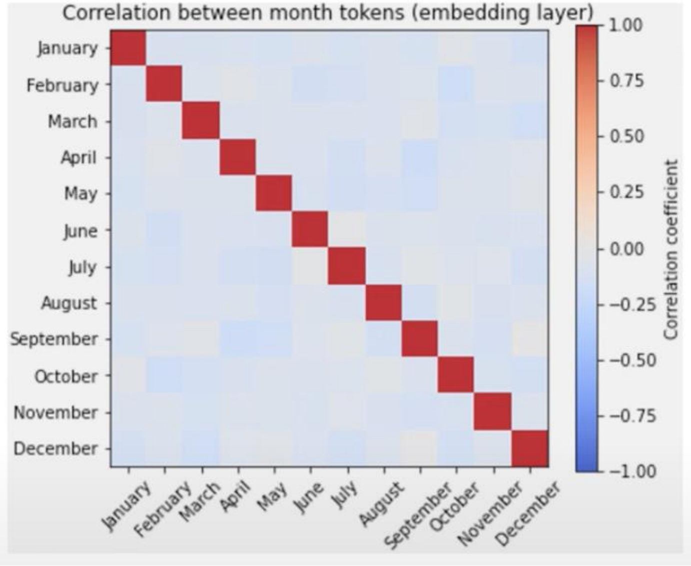
Was Anya Briar Forger born in an even month? *Answer: Yes*.
What is Anya Briar Forger's birth month mod 6? *Answer: 4*.
What is Anya Briar Forger's birth month in numerals? *Answer: 10*.

knowledge comparison

Was Anya Briar Forger born in a month in a year later than Sabrina Eugeo Zuberg? *Answer: [Yes/No]*.
What is Anya Briar Froger's birth month minus Sabrina Eugeo Zuberg's birth month? *Answer: [-11..11]*.

Knowledge Classification and Comparison

				pretrained model				QA finetuned model			
				trained w/o hint	trained with hint			trained w/o hint	trained with hint		
field	task	#train individuals	baseline	test acc	test acc (with hint)	test acc (w/o hint)	hint acc	test acc	test acc (with hint)	test acc (w/o hint)	hint acc
birthmonth	classify %2	(2.5k)	50.0	60.4	77.8	65.2	64.5	61.9	80.4	65.2	69.1
birthmonth	classify %2	(5k)	50.0	67.3	87.3	72.7	80.3	68.0	89.5	72.8	83.9
birthmonth	classify %2	(10k)	50.0	75.9	94.2	80.3	91.0	76.4	95.0	79.9	92.8
birthmonth	classify %2	(25k)	50.0	86.4	98.6	91.1	97.8	87.1	98.8	90.9	98.4
birthmonth	classify %2	(50k)	50.0	95.3	99.5	97.5	99.2	96.3	99.7	97.5	99.5
birthmonth	classify %12	(2.5k)	8.3	51.5	61.5	53.7	61.5	58.3	64.1	53.8	64.0
birthmonth	classify %12	(5k)	8.3	74.2	79.0	70.1	79.0	80.3	82.5	75.0	82.4
birthmonth	classify %12	(10k)	8.3	91.6	92.0	86.8	92.0	93.5	94.7	91.2	94.7
birthmonth	classify %12	(25k)	8.3	97.9	98.5	96.8	98.5	98.9	99.2	98.3	99.2
birthmonth	classify %12	(50k)	8.3	99.4	99.5	99.4	99.5	99.6	99.8	99.7	99.8
birthmonth	ranking	(2.5k)	54.2	53.7	65.4	59.6	44.2	57.3	65.5	57.6	44.9
birthmonth	ranking	(5k)	54.2	59.2	75.5	63.4	63.6	62.5	75.1	63.1	62.6
birthmonth	ranking	(10k)	54.2	65.4	87.7	67.0	82.7	65.9	88.9	66.3	83.9
birthmonth	ranking	(25k)	54.2	75.6	96.7	75.8	95.4	78.3	97.4	72.5	96.3
birthmonth	ranking	(50k)	54.2	85.6	99.0	86.7	98.5	88.6	98.9	82.9	98.3
major	classify %5	(10k)	20.0	23.6	86.4	24.1	84.5	22.8	89.6	23.9	87.9
major	classify %5	(25k)	20.0	24.6	96.7	26.8	96.3	24.8	97.7	27.0	97.2
major	classify %5	(50k)	20.0	31.6	99.3	34.2	99.2	30.0	99.5	33.9	99.4
major	classify %100	(10k)	1.0	30.1	78.7	34.6	79.0	8.9	75.8	22.2	76.1
major	classify %100	(25k)	1.0	79.3	96.0	74.4	96.0	80.0	95.6	77.1	95.3
major	classify %100	(50k)	1.0	91.7	99.0	90.7	99.1	91.8	98.3	92.5	98.1
major	ranking	(10k)	50.5	52.5	88.8	54.1	86.2	52.4	90.3	54.1	88.3
major	ranking	(25k)	50.5	52.2	96.4	53.7	97.3	52.6	96.9	53.6	97.5
major	ranking	(50k)	50.5	53.9	99.6	55.0	99.5	53.6	99.4	55.0	99.3
major	subtraction	(10k)	1.0	1.1	21.6	1.1	82.5	1.0	23.2	1.1	84.3
major	subtraction	(25k)	1.0	1.1	89.1	1.2	96.7	1.2	84.7	1.2	97.0
major	subtraction	(50k)	1.0	1.1	98.4	1.2	99.3	1.1	97.3	1.2	99.0



The model needs 50k data to get accuracy above 95%. However, in the traditional machine learning theorem, we only need 100 data to get almost 100% accuracy.

A pre-trained model means fine-tuned on a normal QA dataset. A QA fine-tuned model means fine-tuning on a classification/comparison QA dataset. This a misleading notation to be honest.

Knowledge Classification and Comparison

knowledge classification

Was Anya Briar Forger born in an even month? *Answer: October; Yes*.
What is Anya Briar Forger's birth month mod 6? *Answer: October; 4*.
What is Anya Briar Forger's birth month in numerals? *Answer: October; 10*.

knowledge comparison

Was Anya Briar Forger born in a month in a year later than Sabrina Eugeo Zuberg? *Answer: October, September; Yes*.
What is Anya Briar Froger's birth month minus Sabrina Eugeo Zuberg's birth month? *Answer: October, September; 1*.

				pretrained model				QA finetuned model			
				trained w/o hint	trained with hint			trained w/o hint	trained with hint		
field	task	#train individuals	baseline	test acc	test acc (with hint)	test acc (w/o hint)	hint acc	test acc	test acc (with hint)	test acc (w/o hint)	hint acc
birthmonth	classify %2	(2.5k)	50.0	60.4	77.8	65.2	64.5	61.9	80.4	65.2	69.1
birthmonth	classify %2	(5k)	50.0	67.3	87.3	72.7	80.3	68.0	89.5	72.8	83.9
birthmonth	classify %2	(10k)	50.0	75.9	94.2	80.3	91.0	76.4	95.0	79.9	92.8
birthmonth	classify %2	(25k)	50.0	86.4	98.6	91.1	97.8	87.1	98.8	90.9	98.4
birthmonth	classify %2	(50k)	50.0	95.3	99.5	97.5	99.2	96.3	99.7	97.5	99.5
birthmonth	classify %12	(2.5k)	8.3	51.5	61.5	53.7	61.5	58.3	64.1	53.8	64.0
birthmonth	classify %12	(5k)	8.3	74.2	79.0	70.1	79.0	80.3	82.5	75.0	82.4
birthmonth	classify %12	(10k)	8.3	91.6	92.0	86.8	92.0	93.5	94.7	91.2	94.7
birthmonth	classify %12	(25k)	8.3	97.9	98.5	96.8	98.5	98.9	99.2	98.3	99.2
birthmonth	classify %12	(50k)	8.3	99.4	99.5	99.4	99.5	99.6	99.8	99.7	99.8
birthmonth	ranking	(2.5k)	54.2	53.7	65.4	59.6	44.2	57.3	65.5	57.6	44.9
birthmonth	ranking	(5k)	54.2	59.2	75.5	63.4	63.6	62.5	75.1	63.1	62.6
birthmonth	ranking	(10k)	54.2	65.4	87.7	67.0	82.7	65.9	88.9	66.3	83.9
birthmonth	ranking	(25k)	54.2	75.6	96.7	75.8	95.4	78.3	97.4	72.5	96.3
birthmonth	ranking	(50k)	54.2	85.6	99.0	86.7	98.5	88.6	98.9	82.9	98.3
major	classify %5	(10k)	20.0	23.6	86.4	24.1	84.5	22.8	89.6	23.9	87.9
major	classify %5	(25k)	20.0	24.6	96.7	26.8	96.3	24.8	97.7	27.0	97.2
major	classify %5	(50k)	20.0	31.6	99.3	34.2	99.2	30.0	99.5	33.9	99.4
major	classify %100	(10k)	1.0	30.1	78.7	34.6	79.0	8.9	75.8	22.2	76.1
major	classify %100	(25k)	1.0	79.3	96.0	74.4	96.0	80.0	95.6	77.1	95.3
major	classify %100	(50k)	1.0	91.7	99.0	90.7	99.1	91.8	98.3	92.5	98.1
major	ranking	(10k)	50.5	52.5	88.8	54.1	86.2	52.4	90.3	54.1	88.3
major	ranking	(25k)	50.5	52.2	96.4	53.7	97.3	52.6	96.9	53.6	97.5
major	ranking	(50k)	50.5	53.9	99.6	55.0	99.5	53.6	99.4	55.0	99.3
major	subtraction	(10k)	1.0	1.1	21.6	1.1	82.5	1.0	23.2	1.1	84.3
major	subtraction	(25k)	1.0	1.1	89.1	1.2	96.7	1.2	84.7	1.2	97.0
major	subtraction	(50k)	1.0	1.1	98.4	1.2	99.3	1.1	97.3	1.2	99.0

Takeaway9:
Even if the model is trained with the chain of thoughts, during inference time, the chain of thoughts is still required.

"what's the birth date of <name> who is a <occupation> and was born in <city>?"
GPT4 accuracy: 99% (selected 4779 celebrities on Wikipedia)

month mod 2

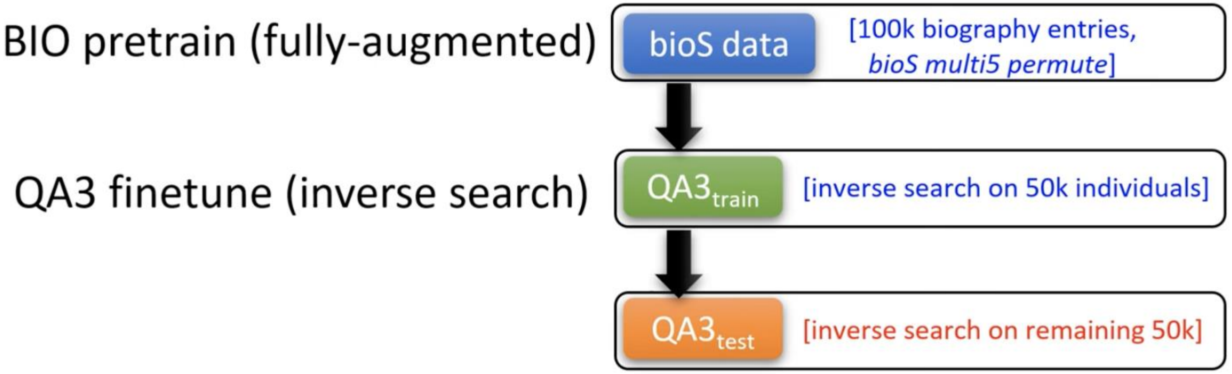
Question: "Answer me yes or no concisely: for <name> who was a <occupation> and was born in <city> in <year>, was this person born in an even month?"
GPT4 correct answer = **50.7%**, incorrect answer = **48.5%,** I don't know = **0.7%**

Question: "Answer me yes or no concisely: was <name1> who was a <occupation1> and was born in <city1> born earlier than <name2> who was a <occupation2> and was born in <city2>?"
GPT4 accuracy answer = **52.3%** among individuals born in 1900~1910
GPT4 accuracy answer = **71.1%** among individuals born in 1900~1950
GPT4 accuracy answer = **81.6%** among all individuals

Rank birth date

GPT4 is not capable of doing this task without the chain of thoughts.

Inverse Search



knowledge inverse search

1. Give me the [first/full] name of the person born on October 2, 1996? (bdate to first, bdate to full)
2. Give me the [first/full] name of the person born on October 2, 1996 in Princeton, NJ? (birth to first, birth to full)
3. Give me the [first/full] name of the person who studied Communications at MIT and worked for Meta Platforms? (three to first, three to full)
4. Give me the [first/full] name of the person who studied Communications at MIT, was born in Princeton, NJ, and worked for Meta Platforms? (four to first, four to full)
5. Give me the [first/full] name of the person who studied Communications at MIT, was born on October 2, 1996 in Princeton, NJ, and worked for Meta Platforms at Menlo Park, CA? (all to first, all to full)

bioS data

Anya Briar Forger was born on October 2, 1996. She spent her early years in Princeton, NJ. She received mentorship and guidance from faculty members at MIT. She completed her education with a focus on Communications. She had a professional role at Meta Platforms. She was employed in Menlo Park, CA.

Inverse Search

	QA bdate_to_first	QA bdate_to_full	QA b_to_first	QA b_to_full	QA all_to_first	QA all_to_full	QA four_to_first	QA four_to_full	QA three_to_first	QA three_to_full
bio7 single	1.0	0.0	1.4	0.0	0.9	0.0	0.9	0.0	1.7	0.0
bio7 single + fullname	0.8	0.0	1.6	0.0	1.4	0.0	1.1	0.0	1.3	0.0
bio7 single + permute1	0.8	0.0	1.4	0.0	1.1	0.0	0.9	0.0	1.3	0.0
bio7 single + permute2	0.8	0.0	1.4	0.0	1.1	0.0	0.6	0.0	1.0	0.0
bio7 single + permute5	0.8	0.0	1.2	0.0	1.4	0.0	1.0	0.0	1.0	0.0
bio7 single + permute1 + fullname	0.6	0.0	1.0	0.0	1.7	0.0	0.9	0.0	1.0	0.0
bio7 single + permute2 + fullname	0.6	0.0	1.4	0.0	1.4	0.0	0.9	0.0	1.0	0.0
bio7 single + permute5 + fullname	0.6	0.0	1.0	0.0	1.1	0.0	0.9	0.0	1.3	0.0
bio7 multi2	0.8	0.0	1.2	0.0	1.4	0.0	0.9	0.0	1.7	0.0
bio7 multi2 + fullname	0.6	0.0	1.2	0.0	1.4	0.0	1.1	0.0	1.7	0.0
bio7 multi2 + permute	0.8	0.0	1.0	0.0	1.7	0.0	1.1	0.0	1.3	0.0
bio7 multi2 + permute + fullname	0.8	0.0	1.2	0.0	1.4	0.0	0.9	0.0	1.3	0.0
bio7 multi5	0.8	0.0	1.4	0.0	1.1	0.0	1.1	0.0	1.0	0.0
bio7 multi5 + fullname	0.6	0.0	1.2	0.0	1.7	0.0	0.8	0.0	1.7	0.0
bio7 multi5 + permute	0.8	0.0	1.4	0.0	1.4	0.0	0.9	0.0	1.7	0.0
bio7 multi5 + permute + fullname	0.6	0.0	1.0	0.0	1.7	0.0	0.9	0.0	1.0	0.0

	MIX bdate_to_first	MIX bdate_to_full	MIX b_to_first	MIX b_to_full	MIX all_to_first	MIX all_to_full	MIX four_to_first	MIX four_to_full	MIX three_to_first	MIX three_to_full
bio7 single	1.0	0.0	1.0	0.0	0.8	0.0	0.6	0.0	0.9	0.0
bio7 single + fullname	1.0	0.0	0.8	0.0	0.6	0.0	1.0	0.0	0.9	0.0
bio7 single + permute1	1.0	0.0	0.8	0.0	0.8	0.0	0.8	0.0	0.7	0.0
bio7 single + permute2	1.2	0.0	0.8	0.0	0.6	0.0	1.0	0.0	1.1	0.0
bio7 single + permute5	0.8	0.0	0.8	0.0	1.0	0.0	0.8	0.0	0.9	0.0
bio7 single + permute1 + fullname	1.2	0.0	1.0	0.0	0.8	0.0	1.0	0.0	0.7	0.0
bio7 single + permute2 + fullname	1.2	0.0	1.0	0.0	0.8	0.0	1.0	0.0	0.9	0.0
bio7 single + permute5 + fullname	1.2	0.0	1.0	0.0	0.8	0.0	1.0	0.0	0.9	0.0
bio7 multi2	1.2	0.0	1.0	0.0	0.8	0.0	1.0	0.0	1.1	0.0
bio7 multi2 + fullname	1.2	0.0	1.4	0.0	1.0	0.0	0.6	0.0	0.9	0.0
bio7 multi2 + permute	1.2	0.0	1.0	0.0	0.8	0.0	0.8	0.0	0.9	0.0
bio7 multi2 + permute + fullname	1.2	0.0	1.0	0.0	1.3	0.0	0.8	0.0	0.9	0.0
bio7 multi5	1.2	0.0	1.0	0.0	0.8	0.0	0.6	0.0	1.1	0.0
bio7 multi5 + fullname	1.2	0.0	1.0	0.0	0.8	0.0	0.8	0.0	0.7	0.0
bio7 multi5 + permute	0.6	0.0	0.8	0.0	1.0	0.0	0.8	0.0	0.9	0.0
bio7 multi5 + permute + fullname	1.0	0.0	1.0	0.0	0.8	0.0	0.8	0.0	0.9	0.0

The models are not able to do the inverse search.

Jane Austen

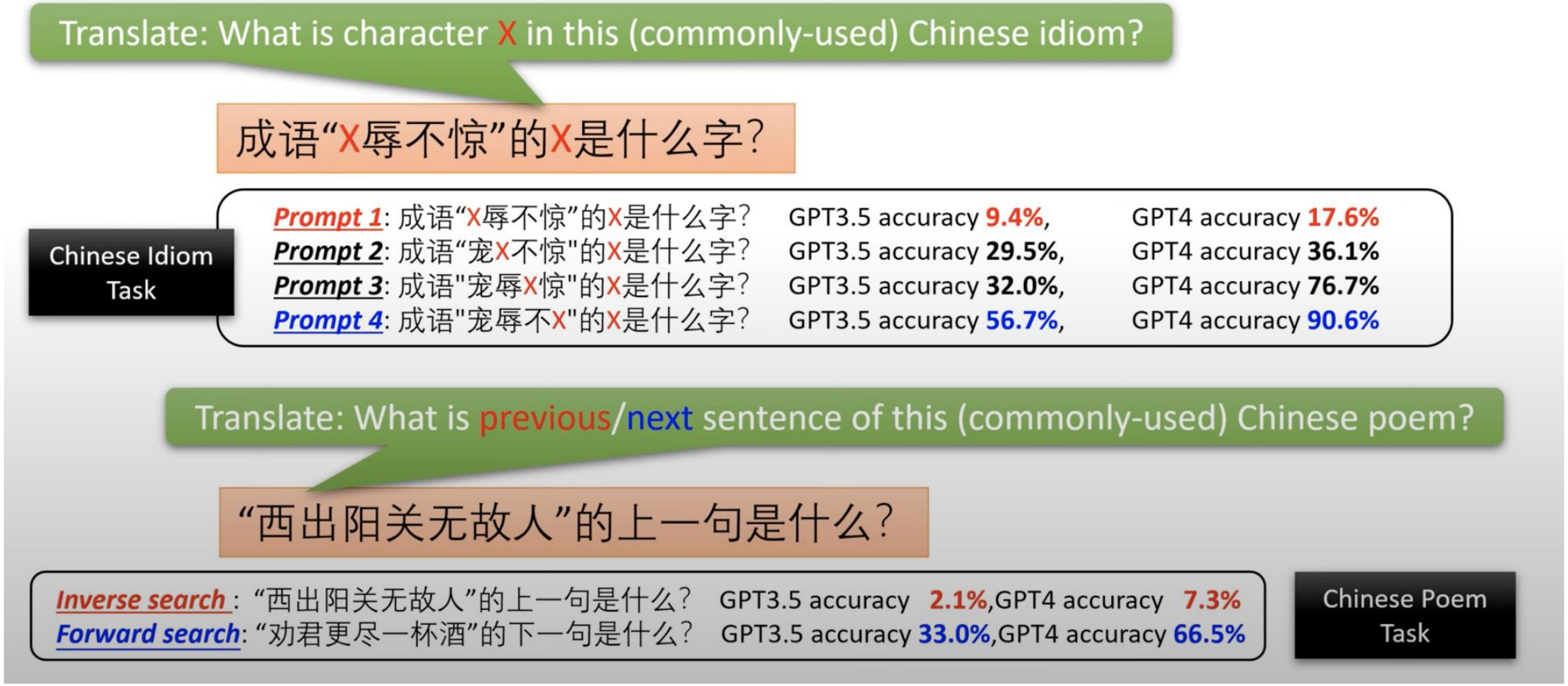
Inverse search: "In <Pride and Prejudice>, what's the sentence ***before***: <sentence2>?"

Forward search: "In <Pride and Prejudice>, what's the sentence ***after***: <sentence1>?"

	Pride & Prejudice	Sense & Sensibility	Persuasion	Northanger Abbey	Emma	Mansfield Park
GPT3.5	0.5% vs 14.4%	0.3% vs 5.4%	0.07% vs 4.3%	0.6% vs 5.5%	0.8% vs 7.2%	0.7% vs 5.5%
GPT4	0.8% vs 65.9%	0.9% vs 40.2%	0.5% vs 33.9%	0.9% vs 41.0%	0.6% vs 42.7%	0.3% vs 31.7%

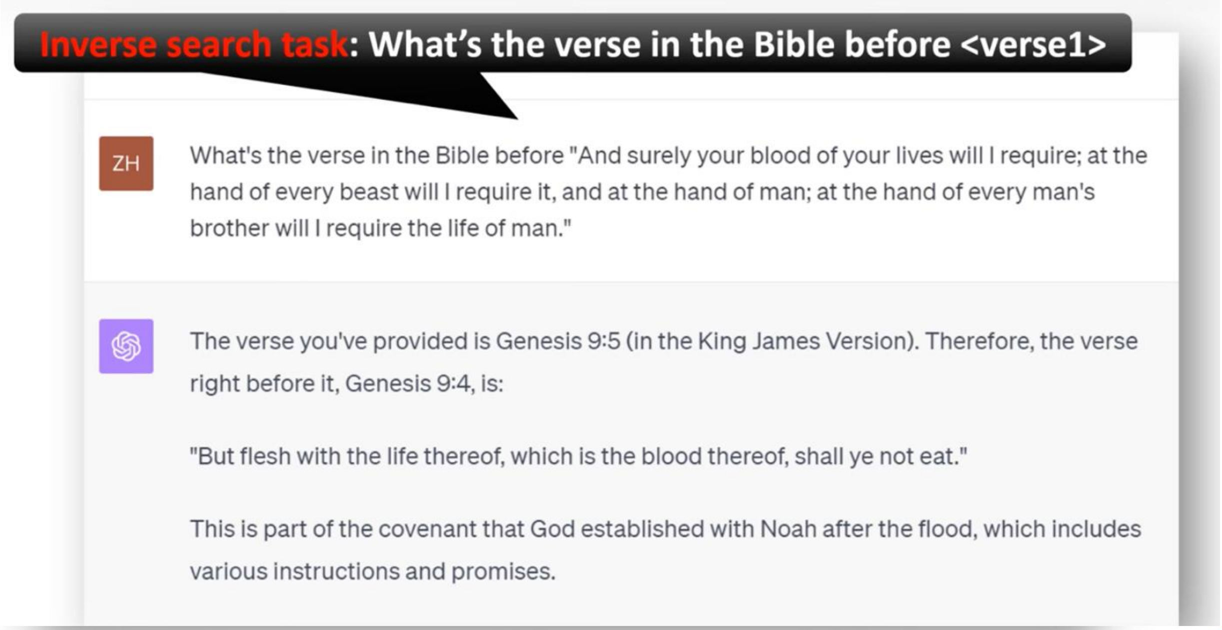
GPT4 is not capable of doing inverse search.

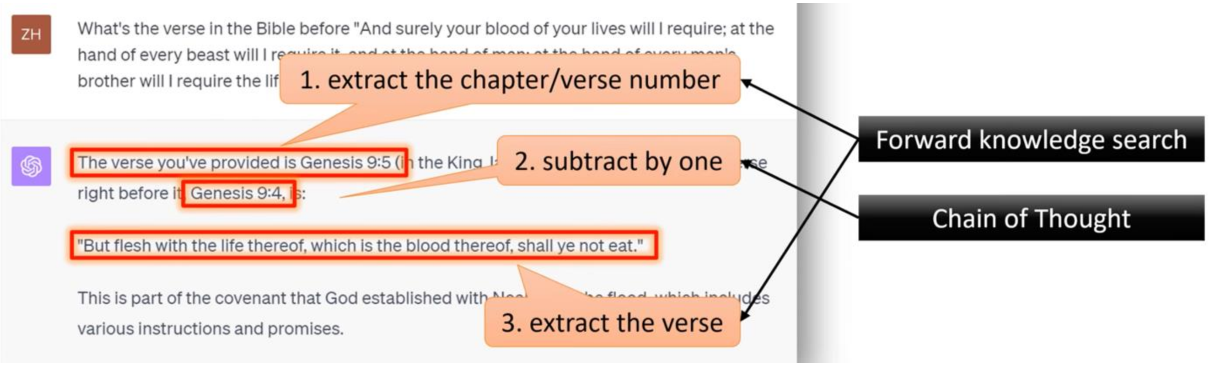
Inverse Search

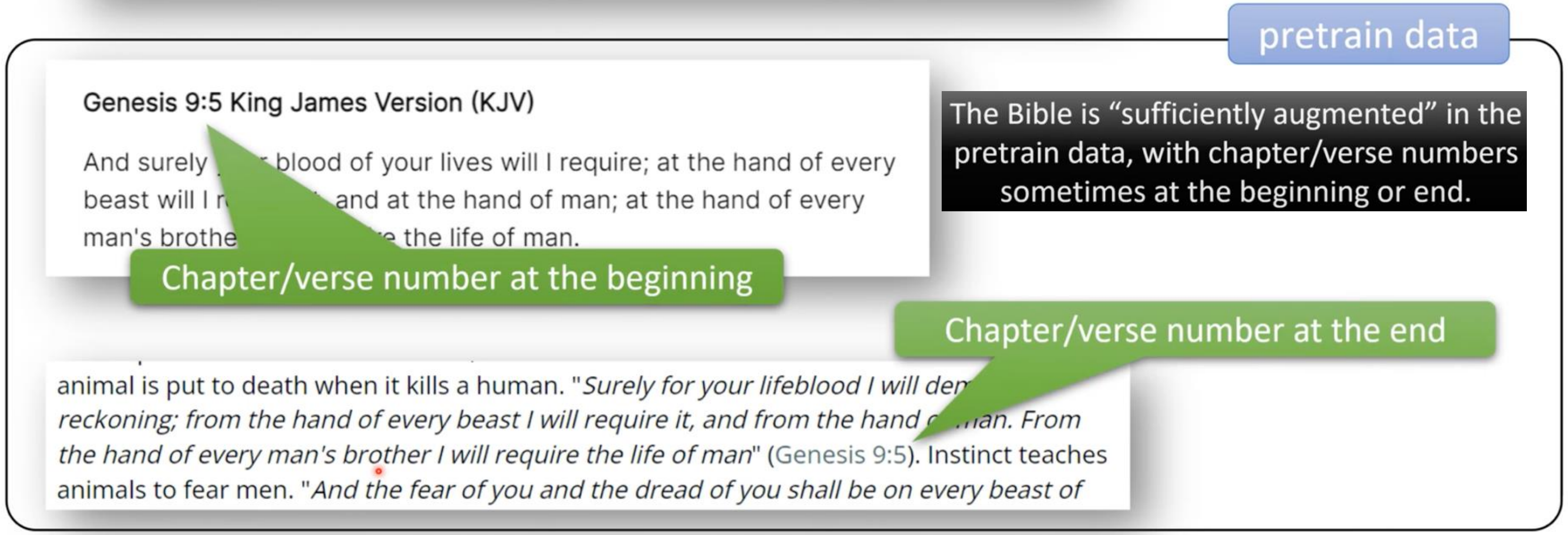


Inverse Search

Takeaway10:
The current model architecture cannot do the inverse search. Therefore, COT or RAG is needed.







Knowledge Capacity Scaling Law

Section 3

Knowledge Definition

- In this paper, a piece of knowledge is a tuple of three strings: (name, attribute, value) = (n, a, v). For instance, n = “Anya”, a = “birthday”, v = “Oct 2, 1996”.

- This format is commonly used in existing literature for research about information extraction, knowledge localization, and knowledge editing etc.

Complexity of Knowledge Set (Motivation)

- The complexity of a knowledge set is determined not only by the **number of knowledge pieces** but also by the **length of the value** string v, the **diversity of the vocabulary**, and other factors. For instance, if the attribute a = “passport number,” then the value v contains more bits of knowledge compared with a = “gender,” because the former has significantly higher **diversity.** If the attribute a = “birth date,” then the value v could consist **of 3 chunks**: (10, 2, 1996).

Complexity of Knowledge Set (Notation)

Considering these examples, we propose a set of hyperparameters that may influence the complexity of knowledge:

Set of names

1. N — the number of (distinct) names n, denoted by $\mathcal{N}$.

|\mathcal{N}| = N

2. K — the number of attributes a, with $\mathcal{A}$ representing the set of attributes. For simplicity, we assume $|\mathcal{A}| = K$ is fixed.

Set of attributes

|\mathcal{A}| = K

3. T — the number of tokens T, where every character in v belongs to $\mathcal{T}$ for some $|\mathcal{T}| = T$. For example, we can think of T as "vocab size" in a tokenizer.

Set of tokens

|\mathcal{T}| = T

Number of chunks for a value

Number of tokens for a chunk

4. C and L — the number of chunks and the length of each chunk for the value: each value $v \in (\mathcal{T}^L)^C$ can be expressed as $v = (v_1, v_2, \cdots, v_C)$, where $v_i \in \mathcal{T}^L$.

5. D — the diversity of chunks: for each piece of knowledge (n, a, v) and $i \in [C]$, the chunk v_i belongs to $\mathcal{D}_a \subset \mathcal{T}^L$, for some set with cardinality $D \stackrel{\text{def}}{=} |\mathcal{D}_a| \ll T^L$.

For example, the birth month can only be from 1 to 12.
So \mathcal{D}_a = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12} \in \mathcal{T}^L with a cardinality much smaller than T^L.
In short, the diversity is defined as the cardinality of the set of all possible values in the chunk.

Knowledge Set Construction (Theoretical)

In our theoretical result, we introduce a dataset $\mathsf{bioD}(N, K, C, D, L, T)$ defined as follows:

Definition 2.2 (bioD data generation). *Consider a fixed set of K attributes, such as a set $\mathcal{A} = \{$"ID 1"... "ID K"$\}$, and a fixed set $\mathcal{N}_0$ of candidate names (with $N_0 \stackrel{\text{def}}{=} |\mathcal{N}_0| \gg N$).*

1. *Generate N names uniformly at random (without replacement) from $\mathcal{N}_0$ to form $\mathcal{N}$*
2. *For each attribute $a \in \mathcal{A}$, generate D distinct strings $w_{1,a}, \cdots, w_{D,a} \in \mathcal{T}^L$ uniformly at random (without replacement) to form the diversity set $\mathcal{D}_a$.*
3. *For each name $n \in \mathcal{N}$ and attribute $a \in \mathcal{A}$, generate value $v^\star(n, a) = (v_1, v_2, \cdots, v_C)$ by sampling each $v_i \in \mathcal{D}_a$ uniformly at random.*

Let $\mathcal{Z} \stackrel{\text{def}}{=} \{(n, a, v^\star(n, a))\}_{n\in\mathcal{N}, a\in\mathcal{A}}$ be the knowledge set.

Step2 D choices for each chunk, sampling from all possible sequences of tokens with length L, so T^L choose D to the power of K

Bit Complexity Upper Bound

Proposition 2.3 (trivial, bit complexity upper bound). *Given $\mathcal{N}_0$ and $\mathcal{A}$ and $\mathcal{T}$, to describe a knowledge set generated in Definition 2.2, one needs at most the following number of bits:*

Step1 N0 choose N

$$\log_2 \binom{|\mathcal{N}_0|}{N} + NKC \log_2 D + K \log_2 \binom{T^L}{D} \approx N \log_2 \frac{|\mathcal{N}_0|}{N} + NKC \log_2 D + KD \log_2 \frac{T^L}{D} .$$

(The approximation is valid when $|\mathcal{N}_0| \gg N$ and $T^L \gg D$.) We will present a bit complexity lower bound in Section 3.

Step3 D choices for each name, each attribute and each chunk, so D^{NKC}

Knowledge Set and Dataset (Empirical Setting)

0. First, middle, and last names from 400, 400, and 1000 choices
1. Birth years range from 1900 to 2099, months 1-12, days 1-28.
2. Birth cities from 200 US cities
3. Universities from a list of 300 US institutions.
4. Majors are from 100 college disciplines
5. Employers are chosen from 263 companies
6. Work location uniquely determined by employer

Anya Briar Forger was born on October 2, 1996. She spent her early years in Princeton, NJ. She received mentorship and guidance from faculty members at Massachusetts Institute of Technology. She completed her education with a focus on Communications. She had a professional role at Meta Platforms. She was employed in Menlo Park, CA. (2.1)

In this paper, we explore three variations of such datasets:

- $\mathsf{bioS}(N)$ represents an online dataset for N individuals, where each biography is generated with new randomness for the *selection* and *ordering* of six sentence templates *on-the-fly*.
- $\mathsf{bioS}^{\mathsf{simple}}(N)$ denotes a similar dataset, but here, each biography is generated once with a fixed random selection and ordering of the sentence templates.
- $\mathsf{bioR}(N)$ refers to the same dataset, but with each biography written 40 times by LLaMA2 [33] to increase realism and diversity.

Bit Complexity Lower Bound (Theoretical)

Consider a model F with weight parameters $W \in \mathcal{W}$. Assume F is trained on a $\mathsf{bioD}(N, K, C, D, L, T)$ dataset $\mathcal{Z}$ as defined in Definition 2.2 using any optimizer; this process is represented as $W = W(\mathcal{Z})$ (the model's weight is trained as a function of the training dataset $\mathcal{Z}$). During the evaluation phase, we express F through two functions: $F^{\top}(W, R)$, which generates names, and $F^{\perp}(W, n, a, R)$, which generates values given (n, a), where R denotes the randomness used in generation. Let $F_1^{\perp}(W(\mathcal{Z}), n, a, R)$ represent the first chunk of $F^{\perp}(W(\mathcal{Z}), n, a, R)$. We evaluate F by calculating the following three cross-entropy losses:[10]

$$\mathbf{loss}_{name}(\mathcal{Z}) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{n \in \mathcal{N}} -\log \mathop{\mathbf{Pr}}_{R} \left[F^{\top}(W(\mathcal{Z}), R) = n \right]$$

Loss on name

$$\mathbf{loss}_{value1}(\mathcal{Z}) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{n \in \mathcal{N}, a \in \mathcal{A}} -\log \mathop{\mathbf{Pr}}_{R} \left[F_1^{\top}(W(\mathcal{Z}), n, a, R) = v_1^{\star}(n, a) \right]$$

Loss on 1st chunk of value

$$\mathbf{loss}_{value}(\mathcal{Z}) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{n \in \mathcal{N}, a \in \mathcal{A}} -\log \mathop{\mathbf{Pr}}_{R} \left[F^{\perp}(W(\mathcal{Z}), n, a, R) = v^{\star}(n, a) \right]$$

Loss on entire value

[10] We use $\mathbb{E}_n$ or $\mathbb{E}_{n,a}$ to denote uniform random selection of $n \in \mathcal{N}, a \in \mathcal{A}$.

Theorem 3.2 (bit complexity lower bound). *Suppose $N \geq \Omega(D \log N)$. We have*

$$\log_2 |\mathcal{W}| \geq \mathop{\mathbb{E}}_{\mathcal{Z}} \left[N \log_2 \frac{N_0 - N}{e^{\mathbf{loss}_{name}(\mathcal{Z})}} + NK \log_2 \frac{D^C}{e^{\mathbf{loss}_{value}(\mathcal{Z})}} + KD \log_2 \frac{T^L - D}{D e^{(1+o(1))\mathbf{loss}_{value1}(\mathcal{Z})}} - o(KD) \right]$$

$$= N \log_2 \frac{N_0 - N}{e^{\mathbb{E}_{\mathcal{Z}} \mathbf{loss}_{name}(\mathcal{Z})}} + NK \log_2 \frac{D^C}{e^{\mathbb{E}_{\mathcal{Z}} \mathbf{loss}_{value}(\mathcal{Z})}} + KD \log_2 \frac{T^L - D}{D e^{(1+o(1))\mathbb{E}_{\mathcal{Z}} \mathbf{loss}_{value1}(\mathcal{Z})}} - o(KD)$$

The goal of the paper is to study how the number of model parameters competes with this bound.

Corollary 3.3 (no-error case). *In the ideal case, if for every data $\mathcal{Z}$, F can generate a name from $\mathcal{N}$ with exact $1/N$ probability each, then $\mathbf{loss}_{name}(\mathcal{Z}) = \log N$; and if F can 100% accurately generate values given (n, a) pairs, then $\mathbf{loss}_{value}(\mathcal{Z}) = \mathbf{loss}_{value1}(\mathcal{Z}) = 0$. In such a case,*

$$\log_2 |\mathcal{W}| \geq N \log_2 \frac{N_0 - N}{N} + NKC \log_2 D + KD \log_2 \frac{T^L - D}{D} - o(KD)$$

asymptotically matching the upper bound Proposition 2.3.

To perfectly store all knowledge, the model needs at least this number of parameters.

Empirical Capacity Ratio

Motivated by Theorem 3.2, ignoring lower order terms, we define the empirical capacity ratio as

Definition 4.1. *Given a model F with P parameters trained over a bioD(N, K, C, D, L, T) dataset $\mathcal{Z}$, suppose it gives $p_1 = \mathbf{loss}_{name}(\mathcal{Z})$, $p_2 = \mathbf{loss}_{value}(\mathcal{Z})$, $p_3 = \mathbf{loss}_{value1}(\mathcal{Z})$, we define its capacity ratio and max capacity ratio*

$$R(F) \stackrel{\text{def}}{=} \frac{N \log_2 \frac{N_0}{e^{p_1}} + NK \log_2 \frac{D^C}{e^{p_2}} + KD \log_2 \frac{T^L}{De^{p_3}}}{P} \; .$$

Correctly extractable knowledge in bits (learned knowledge) / # of parameters

$$R^{\mathsf{max}}(F) \stackrel{\text{def}}{=} \frac{N \log_2 \frac{N_0}{N} + NKC \log_2 D + KD \log_2 \frac{T^L}{D}}{P} \; .$$

All knowledge in bits / # of parameters

For our bioS(N) data, we define a slightly reduced capacity ratio by omitting the diversity term.[11]

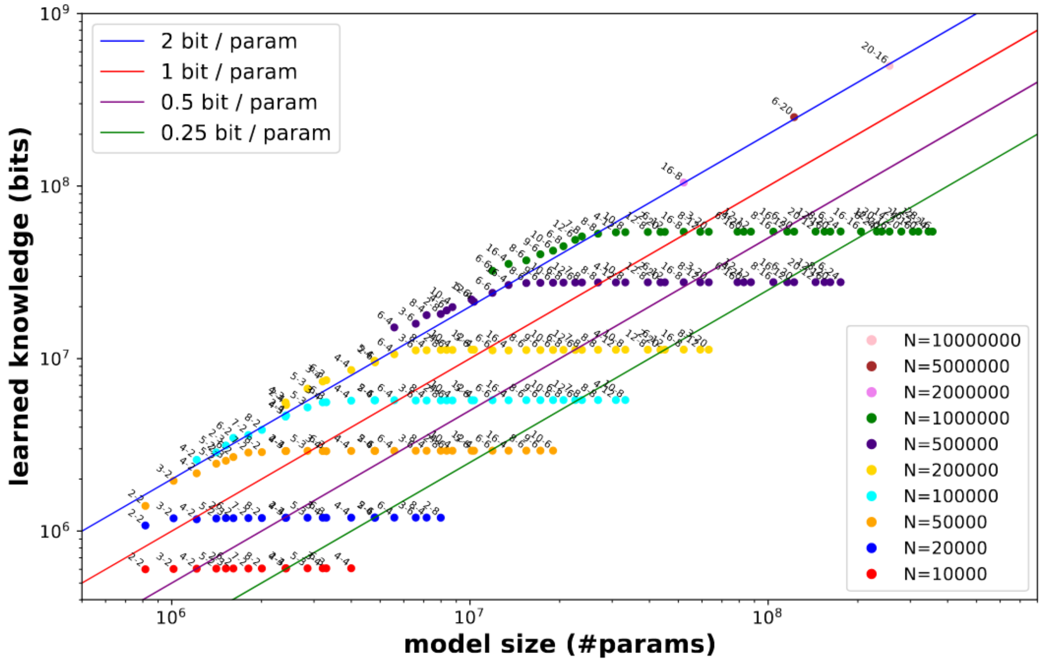
Definition 4.3. *Given a model F with P parameters trained over the bioS(N) dataset $\mathcal{Z}$, suppose it gives $p_1 = \mathbf{loss}_{name}(\mathcal{Z})$ and $p_2 = \mathbf{loss}_{value}(\mathcal{Z})$, its capacity ratio*[12]

$$R(F) \stackrel{\text{def}}{=} \frac{N \log_2 \frac{N_0}{e^{p_1}} + N \log_2 \frac{S_0}{e^{p_2}}}{P} \quad \text{and} \quad R^{\mathsf{max}}(F) \stackrel{\text{def}}{=} \frac{N \log_2 \frac{N_0}{N} + N \log_2 S_0}{P}$$

for $N_0 = 400 \times 400 \times 1000$ and $S_0 = 2 \times (12 \cdot 28 \cdot 200) \times 200 \times 300 \times 100 \times 263$ (c.f. Footnote 9).

Base Scaling Law

The training protocol ensures that each piece of knowledge is presented 1000 times, a process we refer to as “1000 exposures.” It’s important to clarify that this differs from making 1000 passes over the data. For example, a single pass through Wiki data might expose the knowledge (US, capital, Washington D.C.) 1000 times, whereas a pass through the Common Crawl might do so a million times.



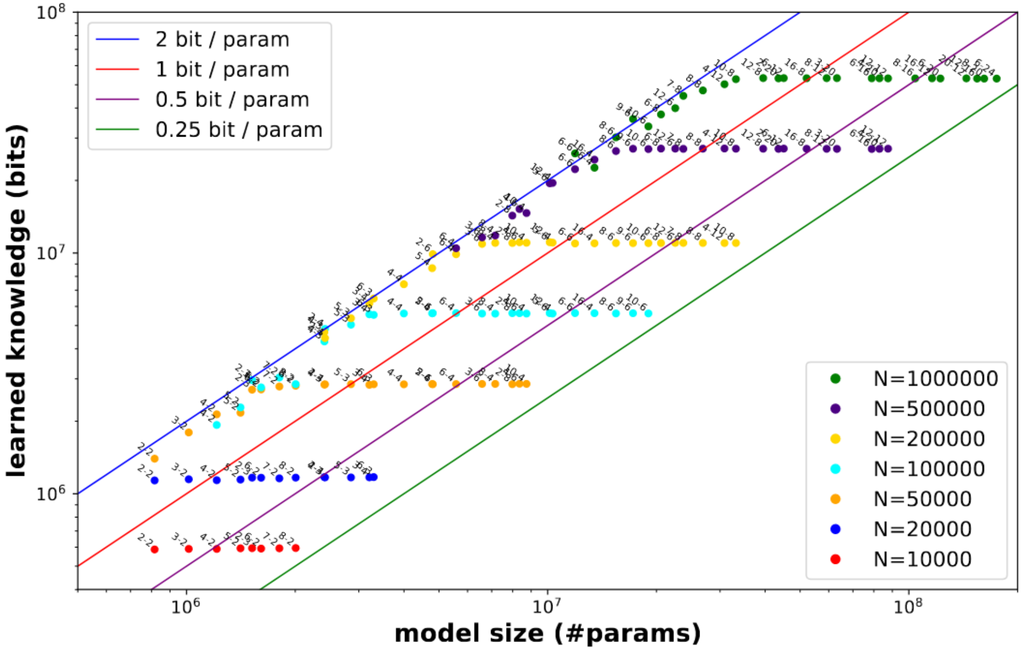
(a) bioS(N) data — **1000 exposures** — peak $R(F) \geq 2$

Takeaway11:

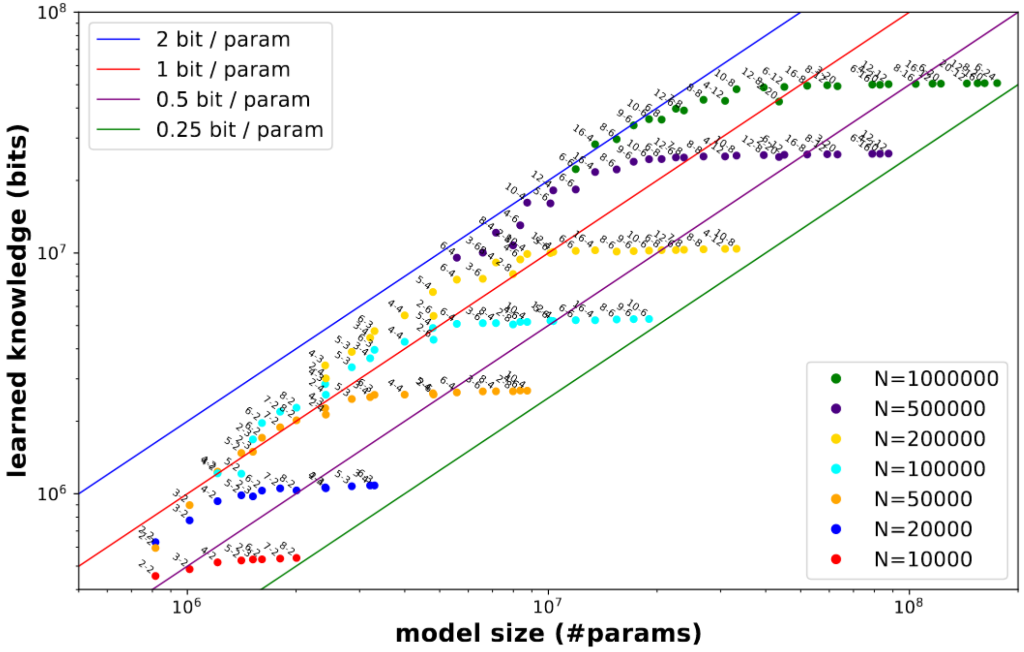
- All models show R(F) > 2 at the peak
- models with Rmax(F) ≤ 1.8 attain near-perfect knowledge accuracies
- across all models, R(F) ≤ 2.3
- In words, this indicates that for a dataset containing B bits of knowledge, selecting a model size P ≥ B/1.8 is sufficient .

Base Scaling Law

Remember BioS simple the dataset with repeated data



(a) on bioS^{simple} data that has no sentence diversity



(b) on the semi-real bioR data generated by LLaMA2

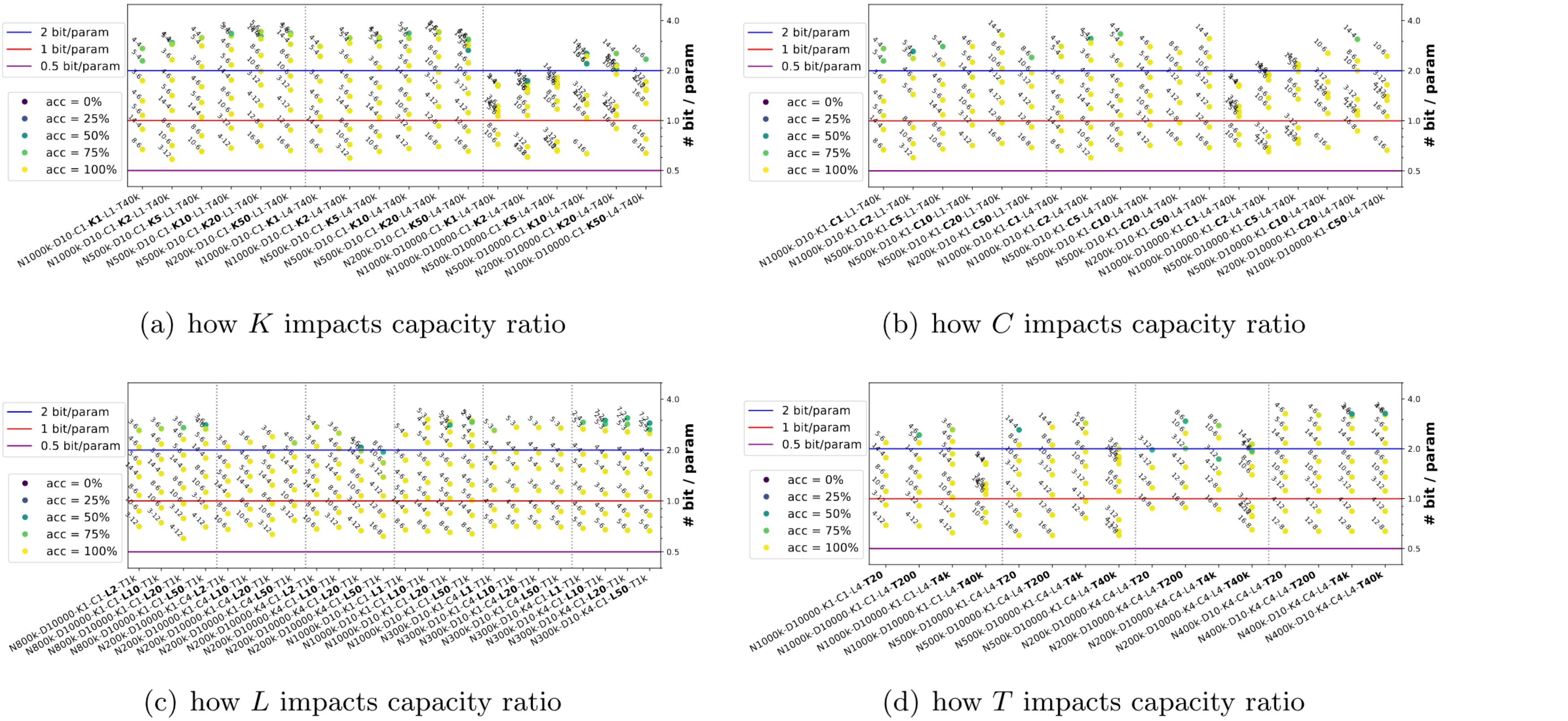
Figure 11: Scaling laws for the bioS^{simple} and bioR data with **1000 exposures**.

Takeaway12:

- In the same 1000-exposure setting, peak capacity ratios for GPT2 trained on bioSsimple and bioR are also approximately 2
- Diverse data (rewriting the same data multiple times) does not hurt — and may sometimes improve — the model's capacity!

Base Scaling Law

Use different variants of BIO dataset to validate the model's capacity is agnostic to dataset



(a) how K impacts capacity ratio

(b) how C impacts capacity ratio

(c) how L impacts capacity ratio

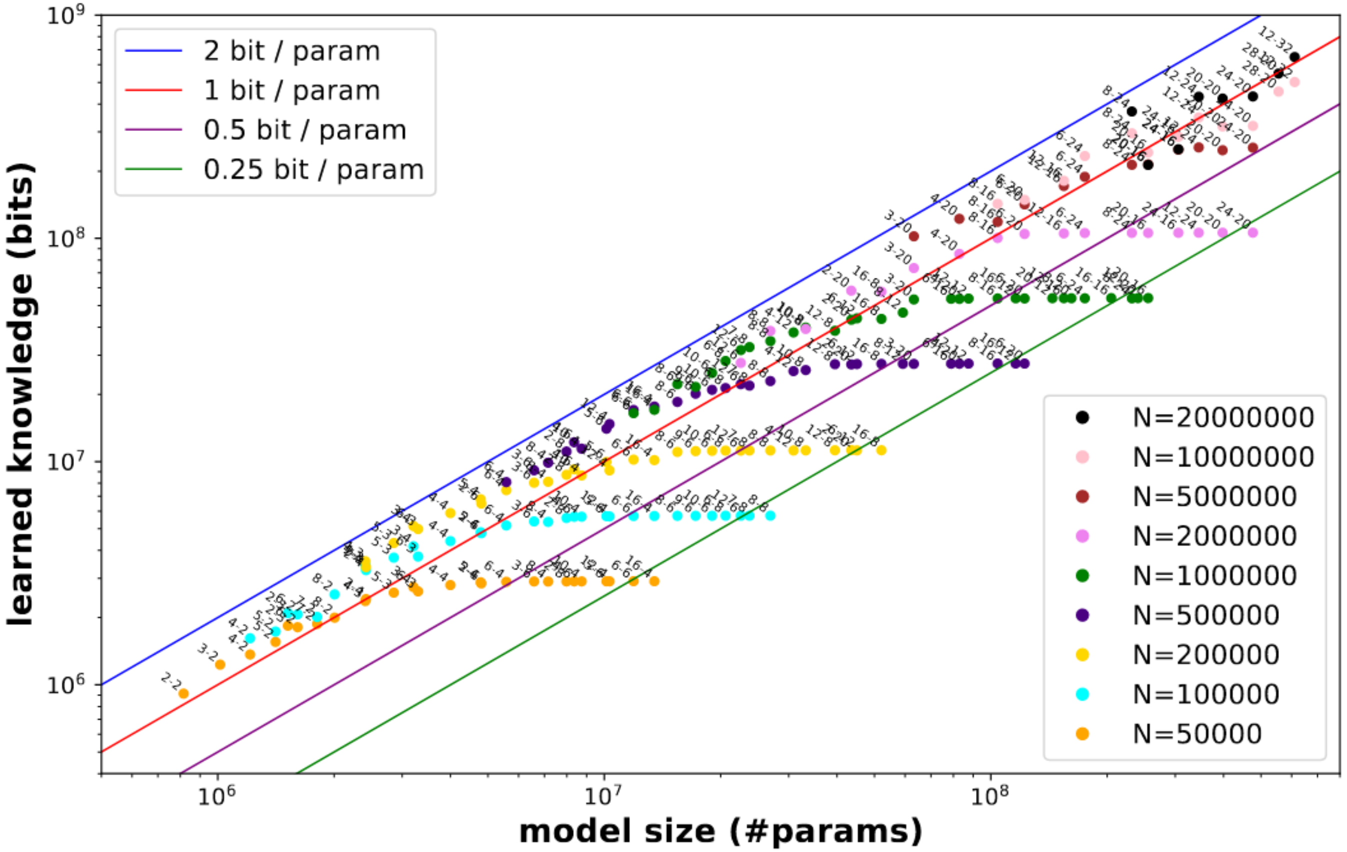
(d) how T impacts capacity ratio

Figure 2: Scaling laws for GPT2 models trained on the bioD(N, K, C, D, L, T) data for **1000 exposures**.

Takeaway13:

- Across a broad spectrum of values, with K, C ranging from 1 to 50, D from 10 to 10, 000, L from 1 to 50, and T from 20 to 40, 000, we observe that: **GPT2 models consistently exhibit a peak capacity ratio R(F) ≥ 2.**

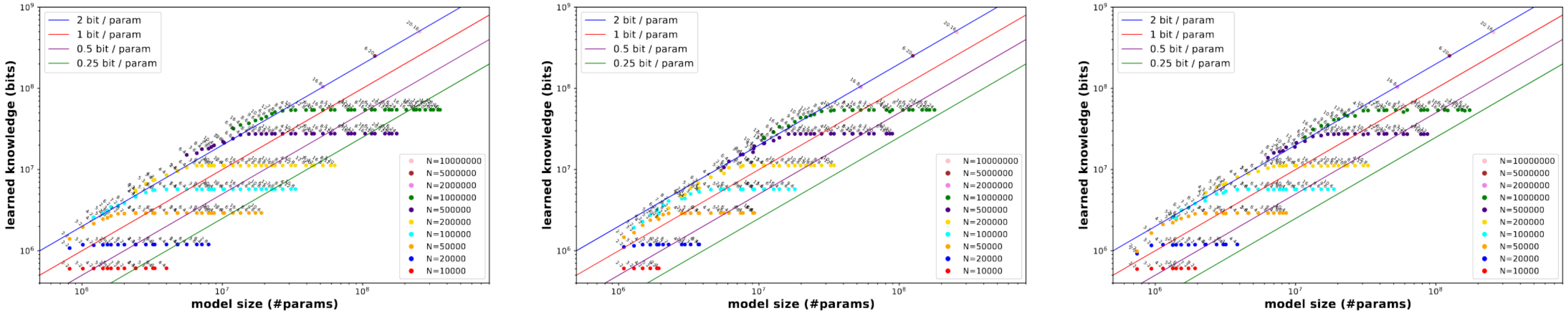
Training Time vs Scaling Law



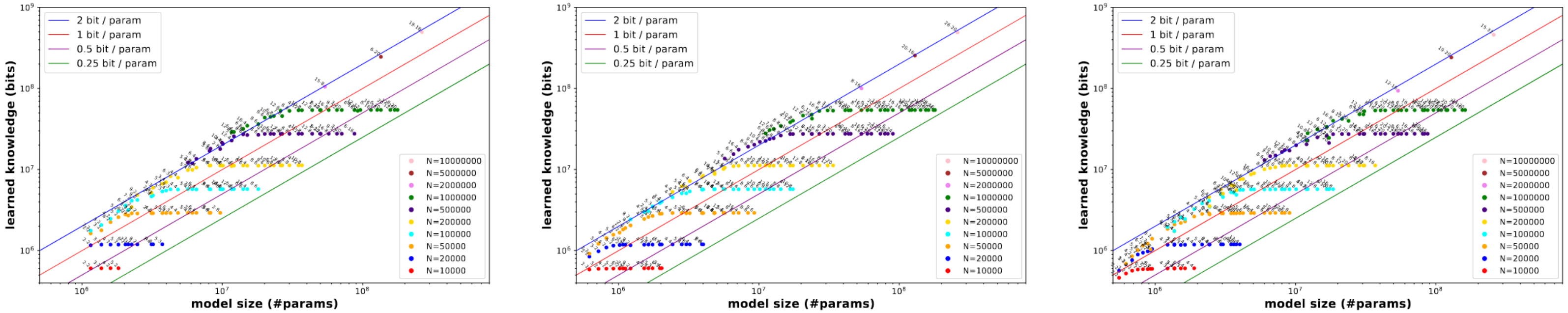
Takeaway14:

- When trained for only 100 exposures on the bioS(N) dataset, with N ranging from 10K to 10M, across a broad spectrum of GPT2 models with sizes from 1M to 0.5B, the peak capacity ratio R(F) consistently exceeds R(F) ≥ 1. **Therefore, although 1000 exposures may be necessary for a model to reach its maximum storage capacity, training with just 100 exposures results in a capacity loss of no more than 2x.**

Model Architecture vs Scaling Law



(a) GPT2, **same** as Figure 1(a) (b) LLaMA (c) LLaMA$^{\text{tied weights}}$



(d) Mistral (e) GPT2 with 1/4-sized MLP (f) GPT2 with no MLP

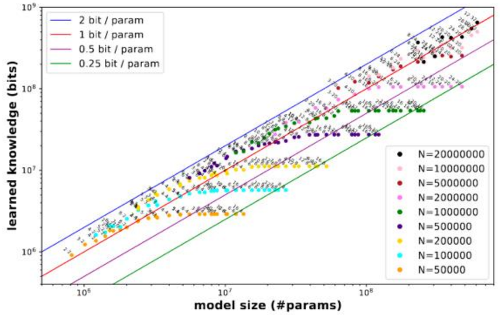
Figure 3: Scaling laws for other model architectures on the bioS(N) data with **1000 exposures**.

This indicates that the 2bit/param capacity ratio is a relatively universal law among most typical (decoder-only) language model architectures.

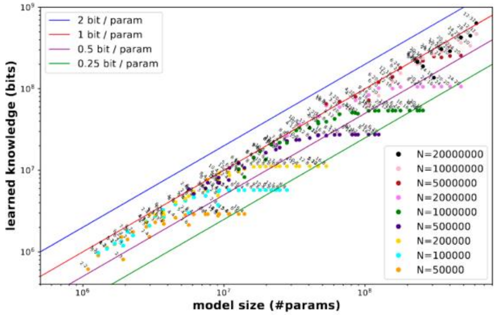
Takeaway15:

- In the 1000-exposure setting, architectures do not matter much:
- LLaMA architecture performs comparably to GPT2
- A similar observation applies to Mistral architecture
- **Reducing the MLP size of GPT2 architecture by 1/4 or even eliminating all MLP layers does not affect its capacity ratio. This suggests, contrary to conventional beliefs, the Attention layers are also capable of storing knowledge .**

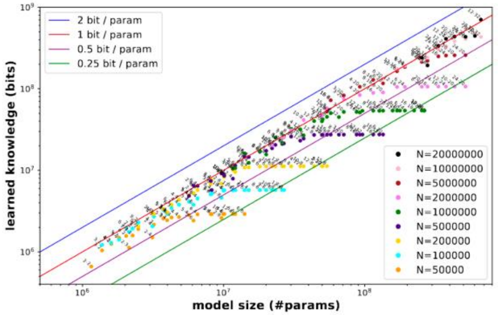
Insufficient Training Regime and a Closer Comparison



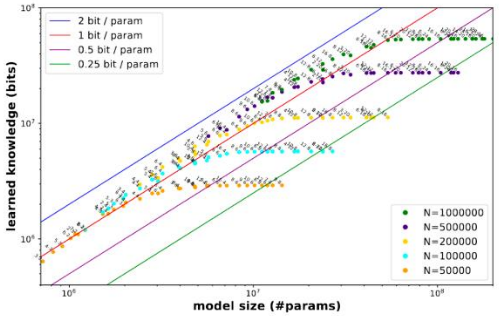
(a) GPT2, **same** as Figure 1(b)



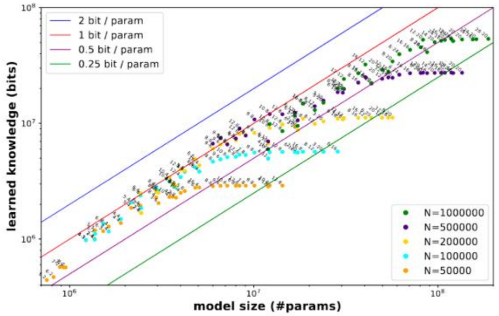
(b) LLaMA



(c) Mistral



(d) GPT2 with 1/4-sized MLP



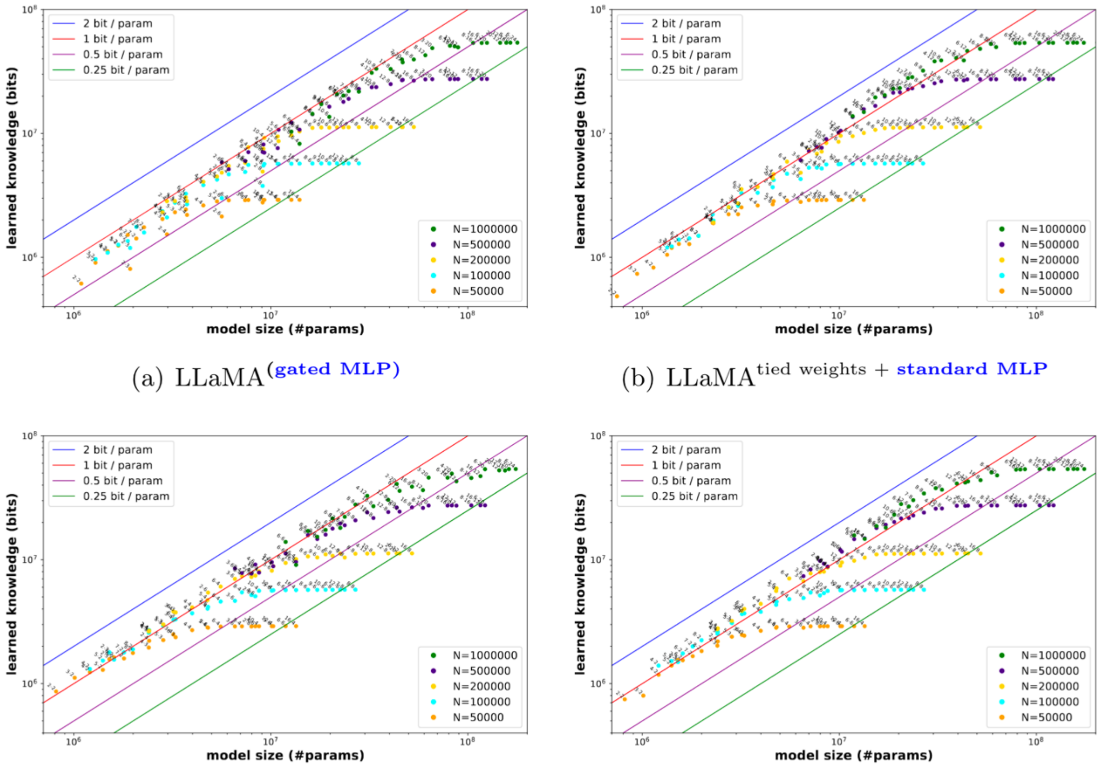
(e) GPT2 with no MLP

Figure 4: Scaling laws for other model architectures on the bioS(N) data with **100 exposures**.

Takeaway16:

- In the 100-exposure setting:
- Even for large models, LLaMA architecture's capacity ratio can be 1.3x worse than GPT2 , even after optimally tuning learning rates. The results are similar for Mistral.
- Reducing GPT2's MLP size by 1/4 has a negligible impact on the capacity ratio.
- Removing MLPs decreases the capacity ratio by more than 1.5x.

Insufficient Training Regime and a Closer Comparison



(a) LLaMA$^{\text{(gated MLP)}}$ (b) LLaMA$^{\text{tied weights + standard MLP}}$

(c) LLaMA$^{\text{tied weights + GPT2Tokenizer (gated MLP)}}$ (d) LLaMA$^{\text{tied weights + GPT2Tokenizer + standard MLP}}$

Figure 5: A closer comparison on LLaMA's scaling laws with bioS(N) data for **100 exposures**.

Takeaway17:

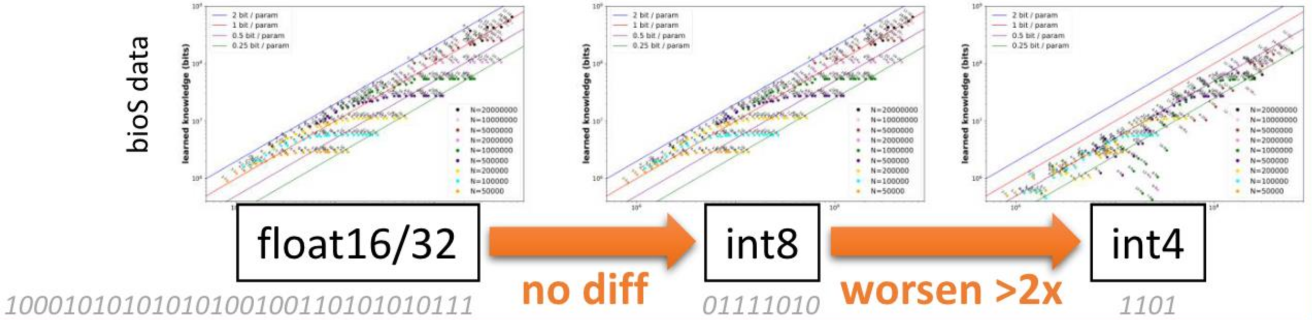
- In the insufficient training regime (notably, the 100-exposure setting), except for tiny models, architectural differences generally do not affect performance, except:
  - Using gated MLP reduces the model's capacity ratio;
  - Removing all MLP layers lowers the model's capacity ratio, although significantly reducing the size of MLPs (e.g., by a 1/4 factor) does not.

Quantization vs Scaling Laws

scaling law (quantization) – Result 8

quantizing → int8 *does not affect* scaling laws *at all*

quantizing → int4 hurts capacity by more than 2x



Takeaway18:

- Quantizing language models (e.g., GPT2) trained with 16-bit floats:
    - to int8 has a negligible impact on their capacity ;
    - to int4 reduces their capacity by more than 2x.

Mixture of Experts vs Scaling Laws

Takeaway19:

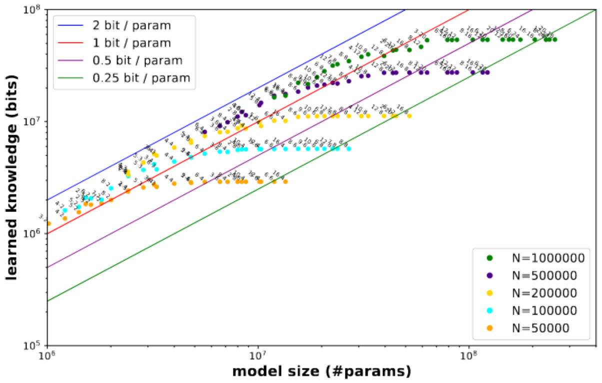
- MoE is nearly fully efficient in storing knowledge , capable of leveraging all its parameters despite the sparsity constraint. Specifically, consider the GPT2-MoE model with 32 experts. If we compute its capacity ratio with respect to the total number of parameters and compare that to GPT2:
  - in the 1000-exposure settings, the peak capacity ratio decreases by 1.3x; and
  - in the 100-exposure settings, the peak capacity ratio decreases by 1.5x.

Junk Data vs Scaling Laws

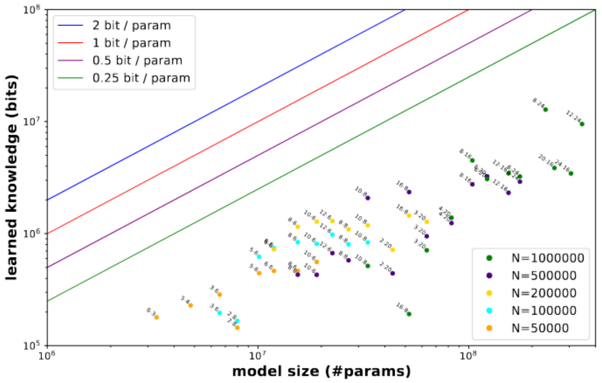
Not all data are useful for knowledge acquisition. For instance, while Wikipedia is full of valuable information, the Common Crawl of web pages may not be (there are also many pieces of information on those webpages, but they may not be useful for a language model to learn, such as the serial number of a random product). How does the presence of low-quality data impact the scaling laws of *useful knowledge capacity*? To investigate this, we create a mixed dataset where:

- $1/8$ of tokens originate from $\mathsf{bioS}(N)$ for various N (referred to as *useful data*), and
- $7/8$ of tokens originate from $\mathsf{bioS}(N')$ for a large $N' = 100M$ (referred to as *junk data*).

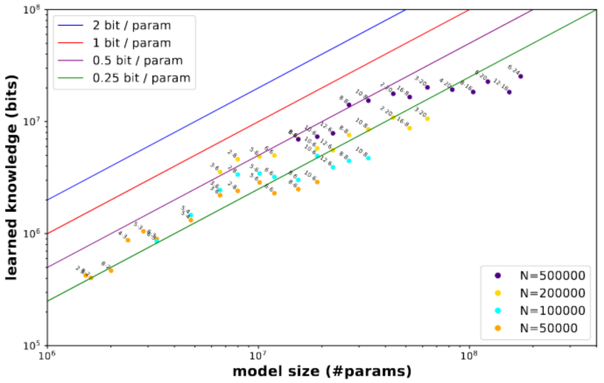
Junk Data vs Scaling Laws



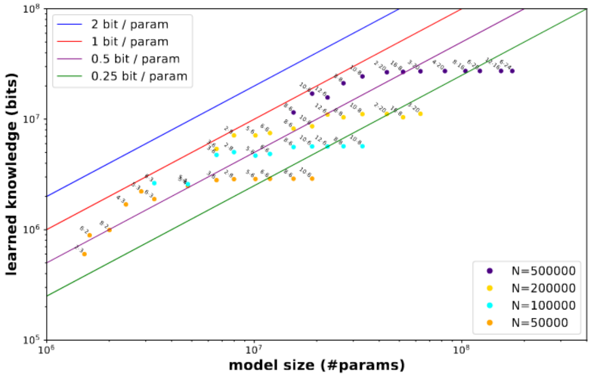
(a) no junk, 100 exposures



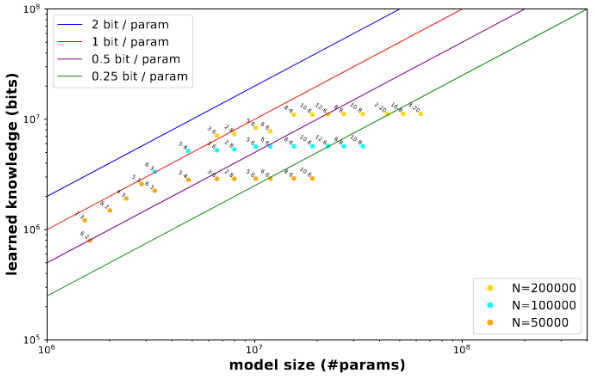
(b) 7/8 junk, 100 exposures



(c) 7/8 junk, 300 exposures



(d) 7/8 junk, 600 exposures

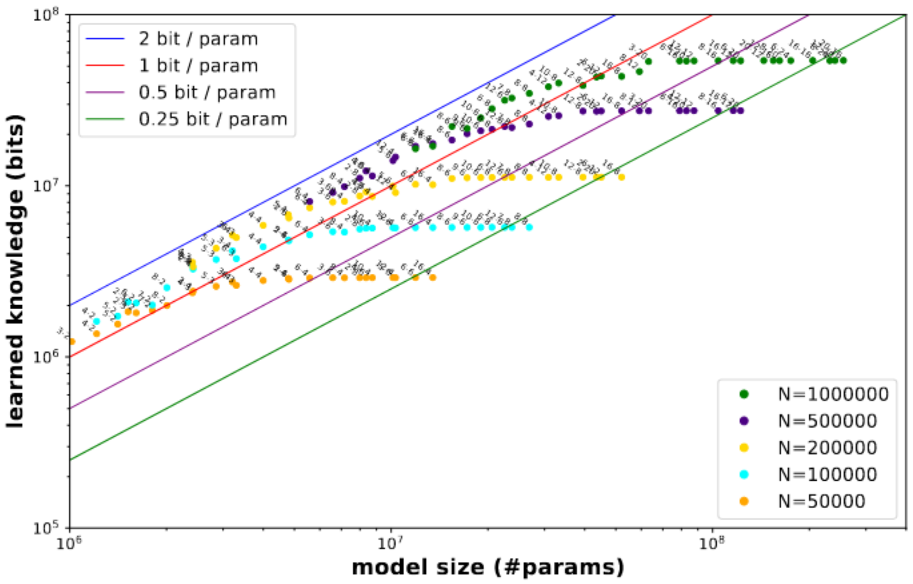


(e) 7/8 junk, 1000 exposures

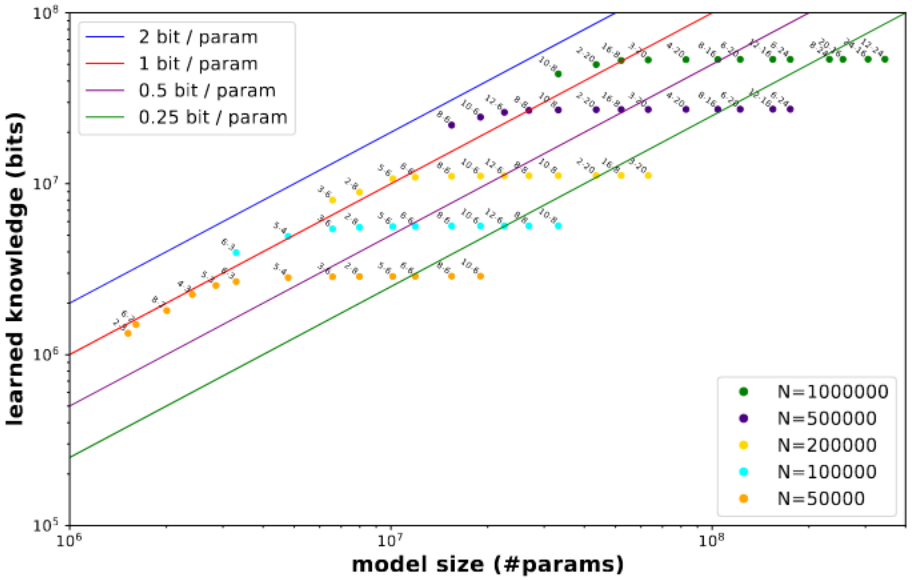
Takeaway20:

- When 7/8 of the training tokens come from junk data (i.e., bioS(N ′) for N ′ = 100M), transformer's learning speed for useful data significantly degrades:
  - If trained for the same 100 exposures, the capacity ratio may degrade by 20x compared with training without junk (compare (b) with (a)).
  - Even trained for 300/600/1000 exposures, the capacity ratio still degrades by 3x/1.5x/1.3x compared with 100 exposures without junk ((c), (d), and (e) vs. (a)).

Junk Data vs Scaling Laws



(a) no junk, 100 exposures

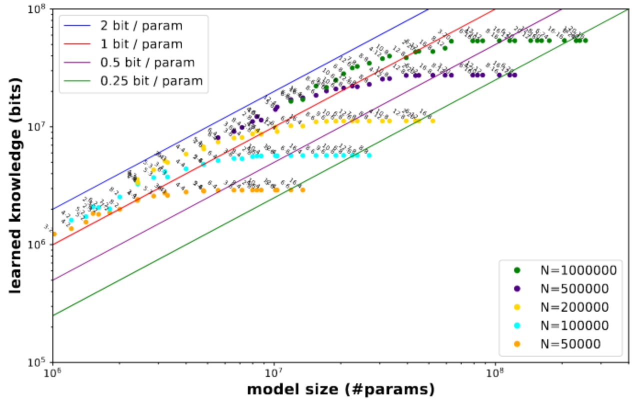


(f) 7/8 **rep-junk**, 100 exposures

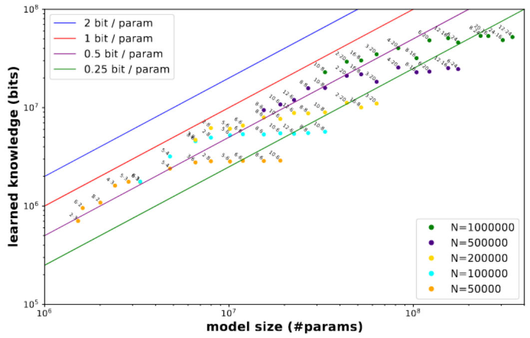
Takeaway21:

- If 7/8 of the training tokens come from highly repetitive data (i.e., bioS(N ′) for N ′ = 1K), this does not affect the learning speed of useful knowledge:
  - The 100-exposure capacity ratio of useful data is unchanged ((f) vs. (a)).

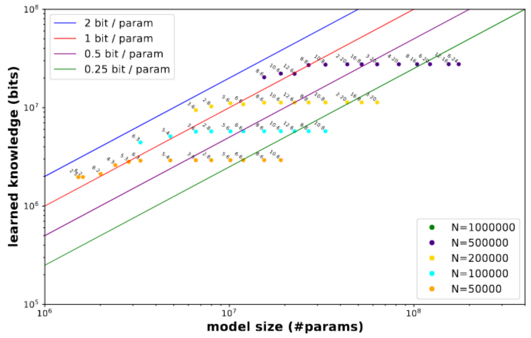
Junk Data vs Scaling Laws



(a) no junk, 100 exposures



(g) 7/8 junk, 100 exposures, **add special symbol**



(h) 7/8 junk, 300 exposures, **add special symbol**

Takeaway22:

- When 7/8 of training tokens are from junk (i.e., bioS(N ′) for N ′ = 100M), adding a special token at the start of every useful data greatly improves capacity ratio:
  - With 100 exposures, the capacity ratio degrades only by 2x ((g) vs. (a)).
  - With 300 exposures, the capacity ratio matches that of the 100-exposure scaling law without junk (compare (h) with (a)).

Summary of Takeaways

1. LLMs don’t have the same natural progression of human knowledge.
2. Fine-tuning cannot help do knowledge extraction if we did not pre-train the model on it.
3. The more augmentation we did for the pre-training dataset, the better results we will have after fine-tuning.
4. If we don’t do the augmentation, the knowledge can only be extracted from the token just before the knowledge. However, if we do the data augmentation, then all knowledge is extractable from the earliest token.
5. If we do the data augmentation, all knowledge is extractable from the person’s name. In other words, “attribute directly saved to the person’s name” is a crucial factor for effective knowledge extraction.
6. Celebrities help minorities.
7. The bidirectional model cannot store all knowledge in a person’s name. Whether the knowledge is stored on the person’s name (pre-train) == QA test accuracy (fine-tune).
8. The necessity of the chain of thoughts.
9. Even if the model is trained with the chain of thoughts, during inference time, the chain of thoughts is still required.
10. The current model architecture cannot do the inverse search.

Summary of Takeaways

11 + 12 + 13. GPT2, trained with standard AdamW, consistently achieves a 2bit/param capacity ratio across all data settings after sufficient training. This includes various model sizes, depths, widths, data sizes, types (synthetic/semi-synthetic), and hyperparameters (e.g., name/value length, attribute number, value diversity).

14. With 100 exposures, an undertrained GPT2's capacity ratio falls to 1bit/param.

15. In the 1000-exposure setting, a 2bit/param capacity ratio appears to be a universal rule: all models, even without MLP layers, closely achieve this ratio.

16. With 100 exposures, some archs show limitations; notably, LLaMA/Mistral's capacity ratio is 1.3x lower than GPT2's, even after best-tuned learning rates.

17. Further controlled experiments indicate that "gated MLP" usage leads to LLaMA/Mistral architecture's underperformance in knowledge storage.

18. Quantizing to int8 does not compromise model capacity (even for models on the boundary of 2bit/param); however, quantizing to int4 reduces capacity to 0.7bit/param.

19. MoE models, even with 32 experts, only reduce 1.3x in capacity compared to the base scaling laws, despite using just 8.8% of the total parameters during inference.

20 + 21. Junk data significantly reduces model capacity. As an example, with a 1:7 ratio of "useful to junk" training tokens, capacity for useful knowledge loses by a factor of 20 x, even when useful knowledge is exposed 100 times.

22. An effective mitigation is to prepend a special token to all useful knowledge. This is akin to adding a domain name like wikipedia.org at the start of every Wikipedia paragraph; the model autonomously identifies high-quality data without prior knowledge of valuable domains.