Skill-it! A data-driven skills framework for understanding and training language models

Mayee F. Chen, Nicholas Roberts, Kush Bhatia, Jue Wang, Ce Zhang, Fred Sala, Christopher Ré

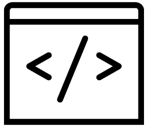
August 31st, Allen AI

mfchen@stanford.edu

together.ai

Motivation

Large language models (LLMs) can do many things:



Write code

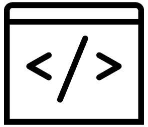
Chat with users

Generate creative content

Why? Architecture, training, **data.** How to select data?

Motivation

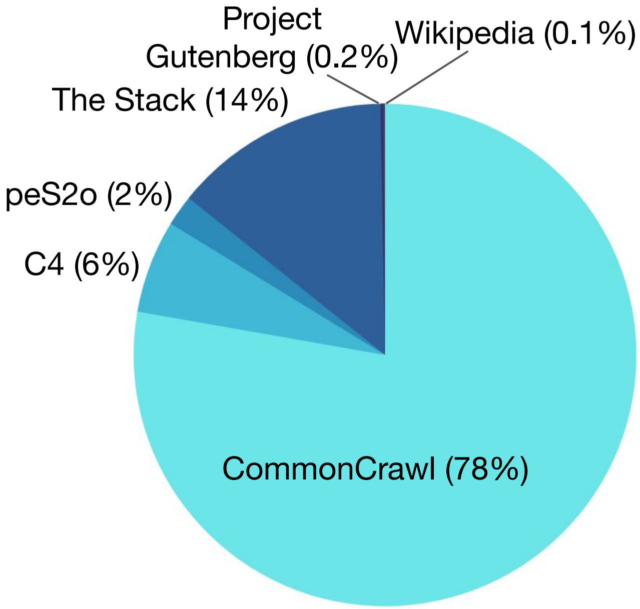
Large language models (LLMs) can do many things:



Write code

Chat with users

Generate creative content



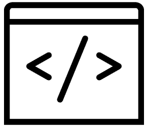
Why? Architecture, training, **data.** How to select data?

Training data mixture[1]

[1] Soldaini et. al. Dolma, 2023.

Motivation

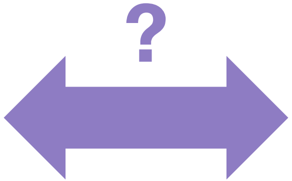
Large language models (LLMs) can do many things:

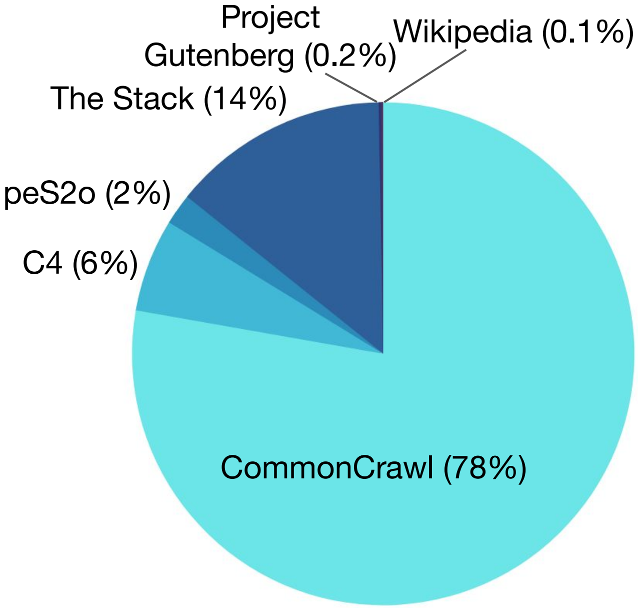


Write code

Chat with users

Generate creative content





Why? Architecture, training, **data.** How to select data?

Training data mixture[1]

[1] Soldaini et. al. Dolma, 2023.

How is training data related to LLM's capabilities?

How do humans learn from data? Learn skills in a certain *order*.

How is training data related to LLM's capabilities?

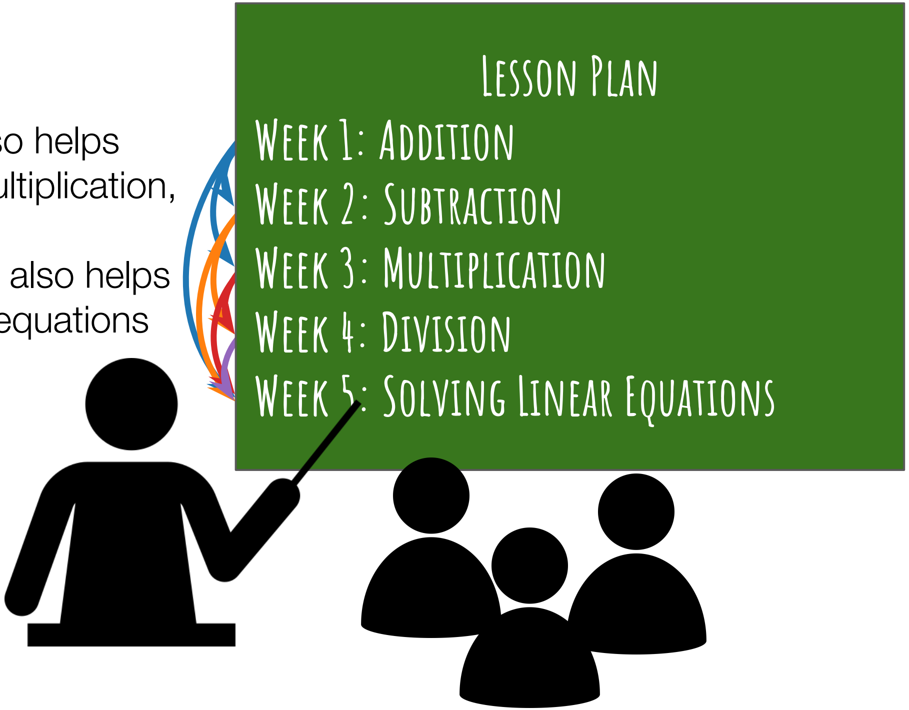
How do humans learn from data? Learn skills in a certain *order.*

How is training data related to LLM's capabilities?

How do humans learn from data? Learn skills in a certain *order.*

Learning hierarchy:

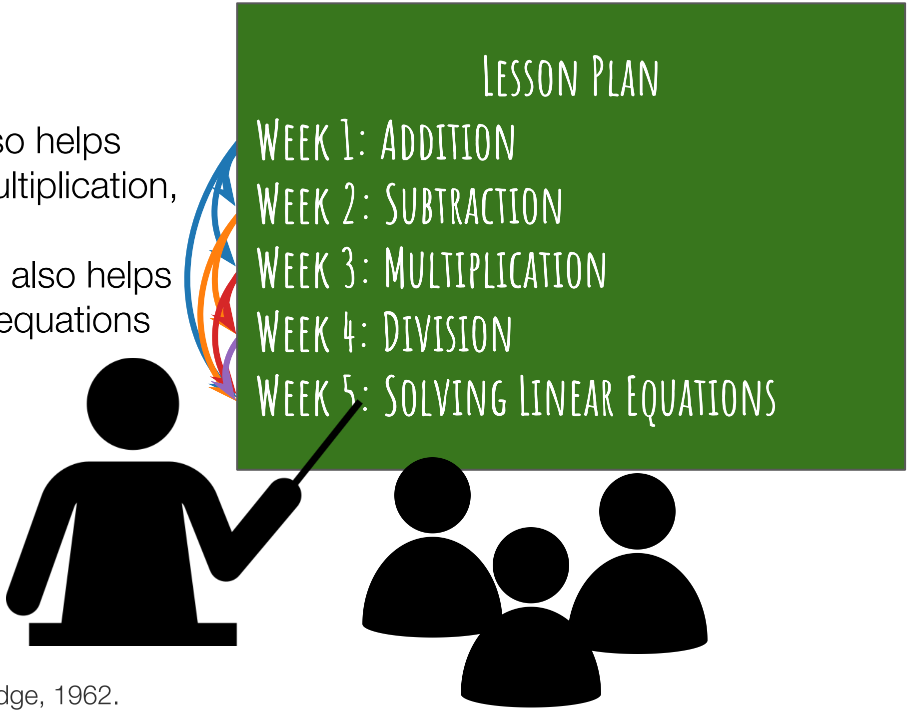
- Learning addition also helps with: subtraction, multiplication, linear equations
- Learning subtraction also helps with: division, linear equations
- And so on



How is training data related to LLM's capabilities?

How do humans learn from data? Learn skills in a certain *order.*

Learning hierarchy:
- Learning addition also helps with: subtraction, multiplication, linear equations
- Learning subtraction also helps with: division, linear equations
- And so on



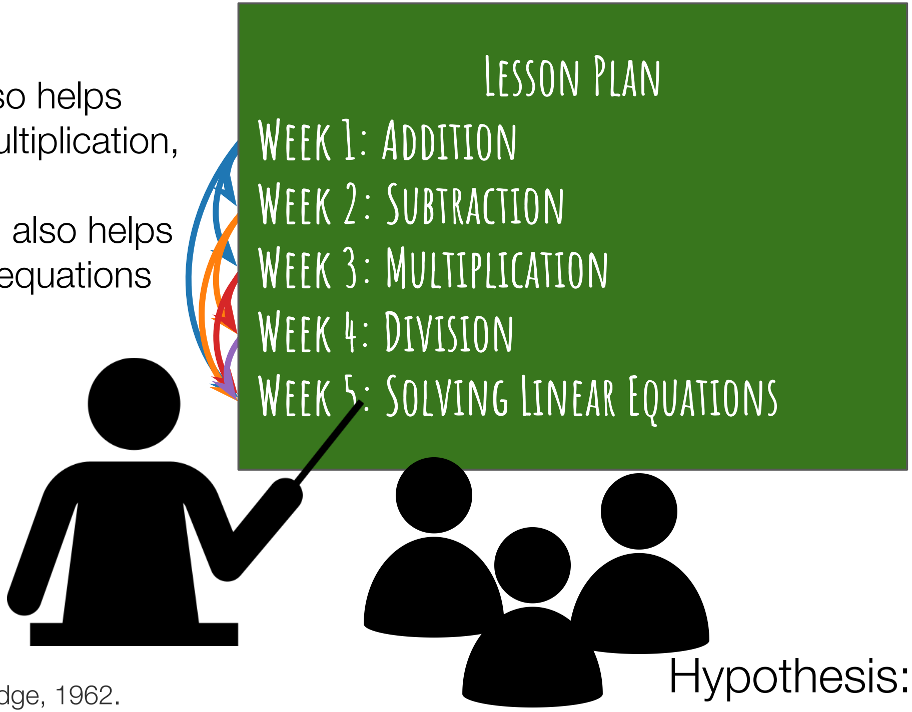
Changing the order of skills makes learning harder![1]
- Learn division then addition → students do worse

[1] Gagne. The acquisition of knowledge, 1962.

How is training data related to LLM's capabilities?

How do humans learn from data? Learn skills in a certain *order.*

- Learning addition also helps with: subtraction, multiplication, linear equations
- Learning subtraction also helps with: division, linear equations
- And so on



Changing the order of skills makes learning harder![1]

- Learn division then addition → students do worse

Hypothesis: models also learn like this

[1] Gagne. The acquisition of knowledge, 1962.

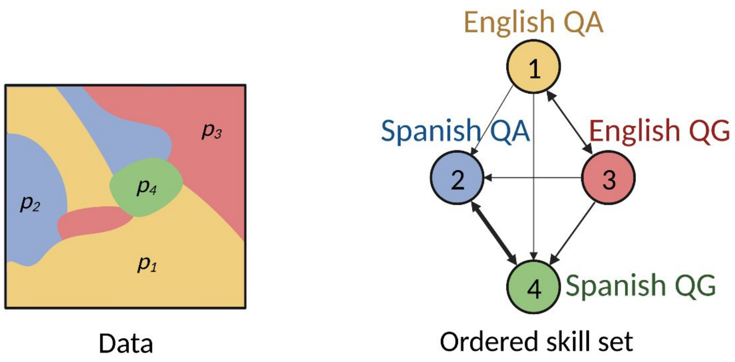
Q: How does training data influence various model capabilities?
Can we use this understanding to more effectively select data to improve such capabilities?

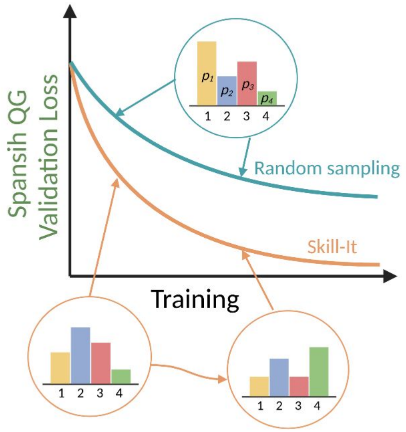
Q: How does training data influence various model capabilities?
Can we use this understanding to more effectively select data to improve such capabilities?

A: there exist sets of *skills* associated with data that the LLM learns most efficiently in some particular order. We can learn this order and exploit it to better select training data.

Outline

- Defining ordered skill sets
- Skill-It data selection algorithm
- Results
- Discussion





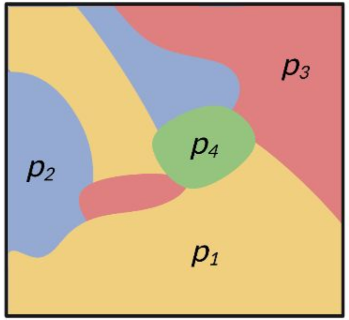
Defining ordered skill sets

Definitions: what is a skill?

Training data can be partitioned into subsets associated with *skills.*

Def (informal): a **skill** s is a unit of behavior with associated data X_s s.t. if a model f is trained on dataset $D_s \subset X_s$, f has improved metric L (e.g., validation loss) on samples belonging to $X_s \setminus D_s$ on average.

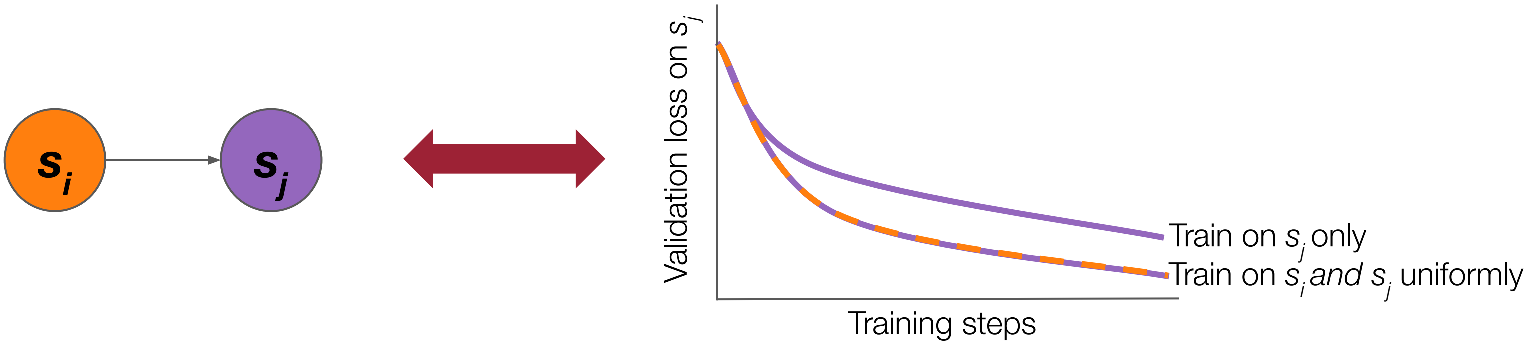
Examples: tasks, data sources, task categories.



Data

How do we define a meaningful order over skills?

Def: Given a set of skills S, its **skills graph** is $G = (S, E)$, where given a fixed data/training budget, $s_i \rightarrow s_j \in E$ iff validation loss on s_j when trained on mixture of s_i and s_j is no greater than when trained on just s_j.

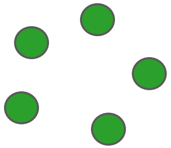


How do we define a meaningful order over skills?

Def: an **ordered skills set** is a set of skills whose **skills graph** is neither empty nor complete.

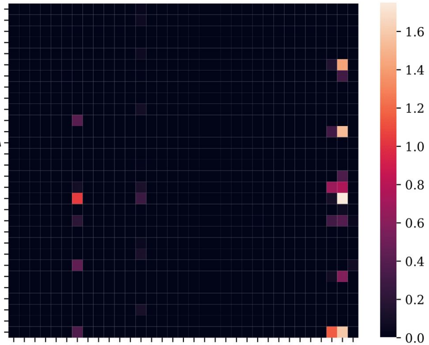
How do we define a meaningful order over skills?

Def: an **ordered skills set** is a set of skills whose **skills graph** is neither empty nor complete.



Dataset/skills: Pile of Law[1]/data sources

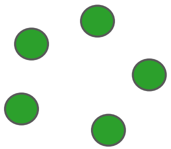
Skills graph's adjacency matrix A

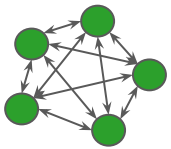


Ordered skill set? ✗

How do we define a meaningful order over skills?

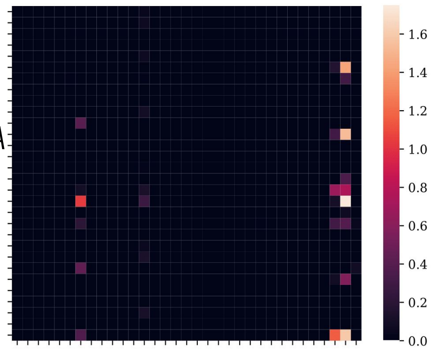
Def: an **ordered skills set** is a set of skills whose **skills graph** is neither empty nor complete.

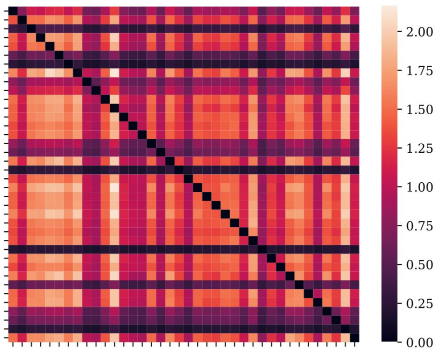




Dataset/skills: Pile of Law[1]/data sources | Alpaca[3]/leading verb

Skills graph's adjacency matrix A

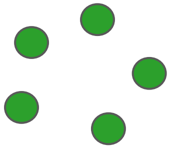


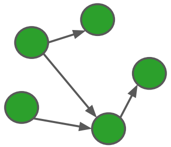


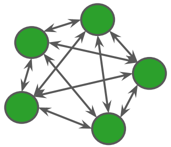
Ordered skill set?

How do we define a meaningful order over skills?

Def: an **ordered skills set** is a set of skills whose **skills graph** is neither empty nor complete.

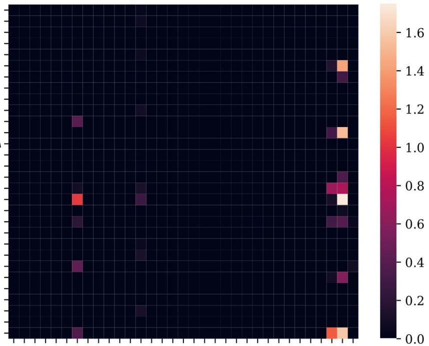


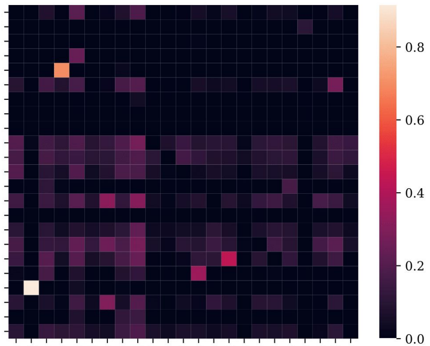


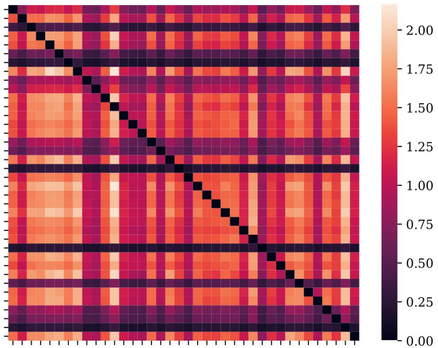


Dataset/skills: Pile of Law[1]/data sources | Natural Instructions[2]/task categories | Alpaca[3]/leading verb

Skills graph's adjacency matrix A





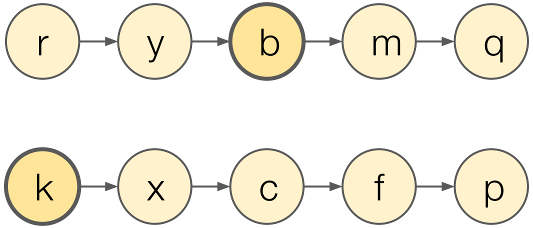


Ordered skill set?

[1] Henderson et. al., 2022.
[2] Wang et. al., 2022.
[3] Taori et. al., 2023.

Ordered skill set examples: LEGO synthetic[1]

Input	Output
b = not y, r = val 1, m = val b, q = val m, y = not r. b = ?	1
c = val x, p = val f, x = val k, f = not c, k = val 0. k = ?	0



Skill 3

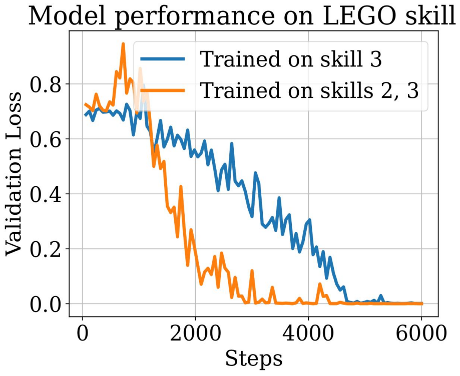
Skill 1

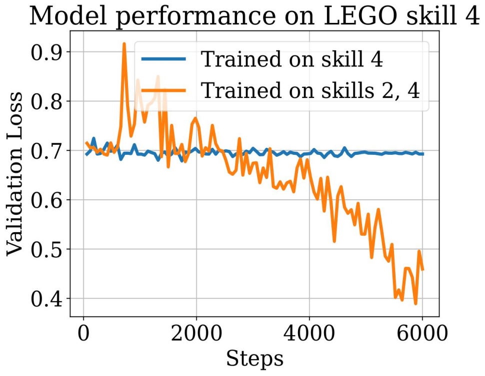
- Each sample is a series of recursive variable assignments to 0 or 1. The model is asked to provide the value of a variable.
- Skill X = model's ability to get the value of Xth variable correct
  - Skill 1 = recall only, easy
  - Skill *k* = need to learn skill *k-1*, harder
- Intuitive guess: skills graph is a "chain"

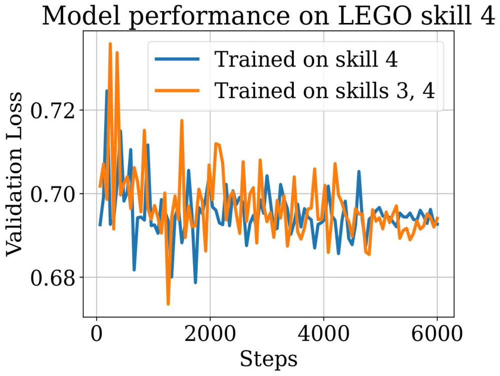
[1] Zhang et. al., 2022.

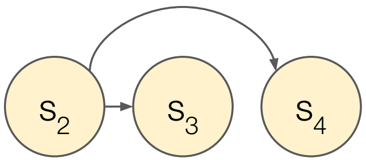
Ordered skill set examples: LEGO synthetic

- Continually pre-train GPT-Neo 125M on concatenated LEGO input/output pairs
- Measure loss on output token for held-out validation set



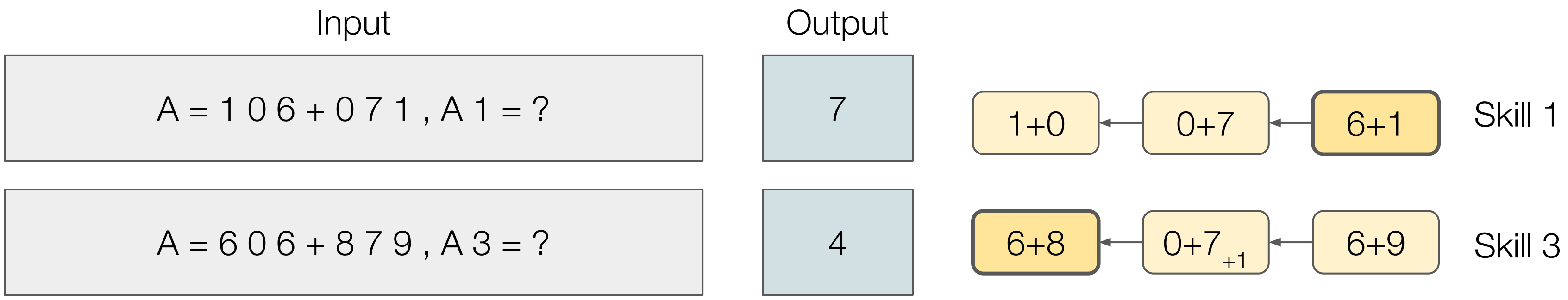






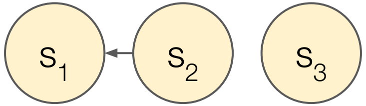
There is an ordered skill set, but not the one we guessed!

Ordered skill set examples: Addition synthetic



- Skill X = model's ability to get the Xth digit of the addition correct
  - Skill 1 = no carry, easy
  - Skill *k* = may need carry/dependent on skill *k-1*, harder
- Intuitive guess: skills graph is a "chain"

Ordered skill set examples: Addition synthetic



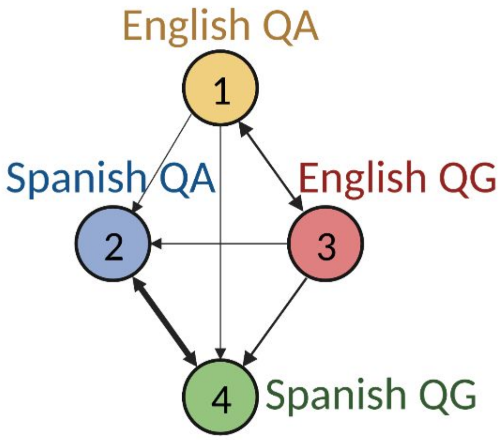
There is an ordered skill set, but not the one we guessed!

Ordered skill set examples: Natural Instructions

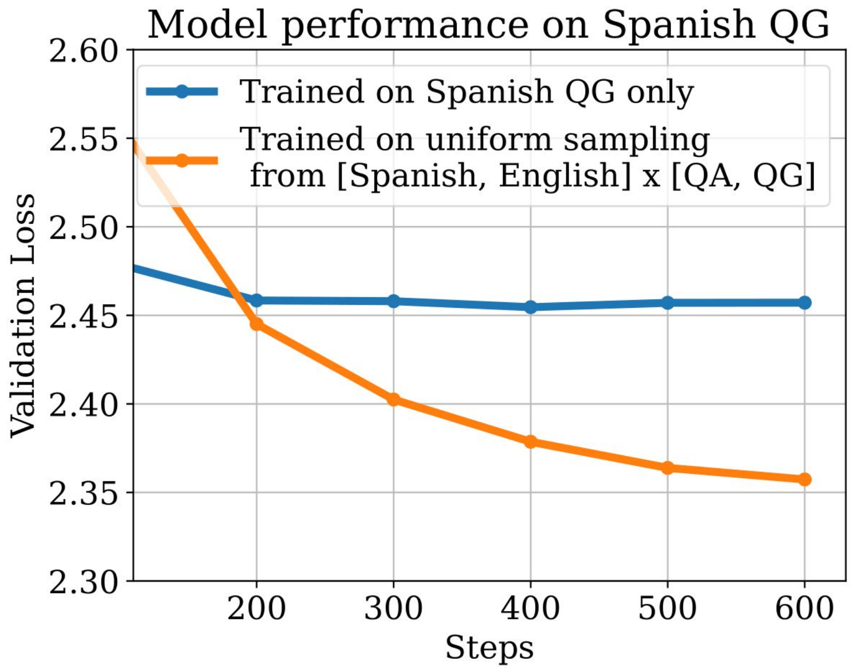
<u>Skills</u>

English question answering	Input: 3sat began broadcasting on 1 December 1984. What year did 3sat start? Output: 1 december 1984
English question generation	Input: Snow Wolf is an espionage novel by Glenn Meade. Output: Who wrote the Snow Wolf
Spanish question answering	Input: Que inventen ellas es el título de la muestra sobre científicas e investigadoras, que estará hasta el 4 de enero en la Obra Social de Caja Madrid, en la Plaza de Catalunya. ¿Dónde estará disponible la muestra? Output: en la Obra Social de Caja Madrid
Spanish question generation	Input: O sea, lloraba el músico. Volverá en octubre Pedrito. Volverá, ya lo verán. Volverá. Dios mío. Output: ¿Cuándo volverá Pedrito?

Ordered skill set examples: real data



Ordered skill set



Ordered skill set examples: real data

<u>Skills</u>

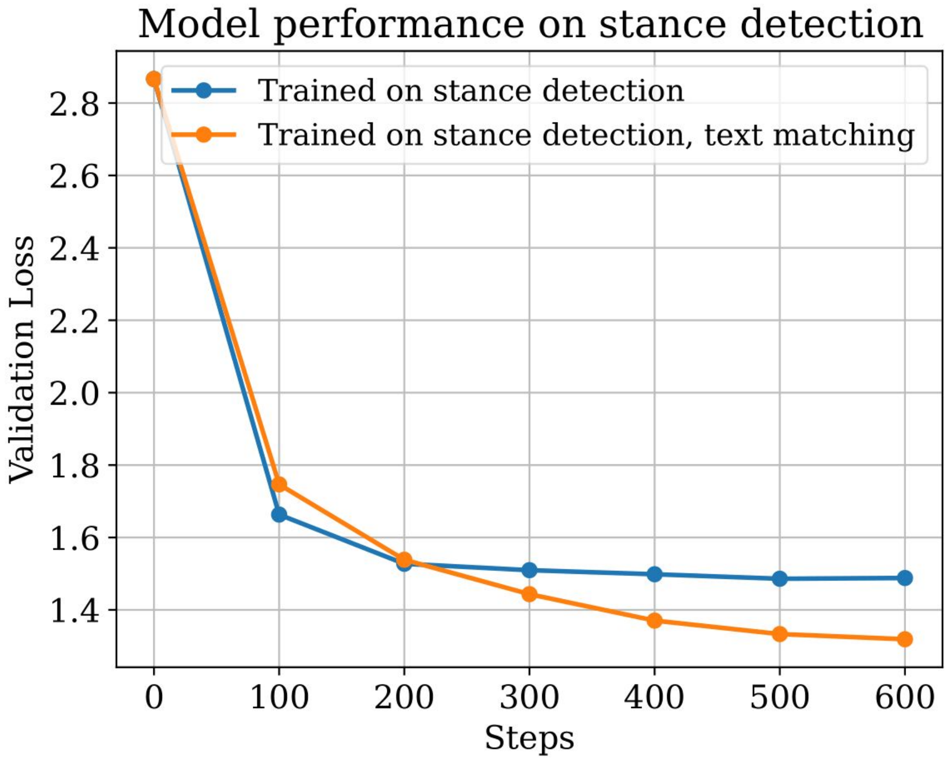
Stance Detection

Input:
Topic: Airport security profiling
Argument: Profiling will help avoid invasive scanners and pat-downs.
Output: in favor

Text Matching

Input:
Sentence 1: What are considered the safest prescription pills for high blood pressure/hypertension?
Sentence 2: What are the common side effects of blood pressure medications?
Output: Dissimilar

Ordered skill set examples: real data



Skill-It data selection algorithm

Problem Setup

Input:

- Training skill set $S_{train} = s_{train,1}, \ldots, s_{train,k}$ with associated training corpus $X_{train} = X_{train,1}, \ldots, X_{train,k}$
- Evaluation skill set $S_{eval} = s_{eval,1} \ldots, s_{eval,m}$ with associated held-out validation data X_{eval}
- Budget of *n* samples
- Pre-trained LLM *f*

Goal: how to order/select *n* samples from X_{train} for *f* to perform well on X_{eval}

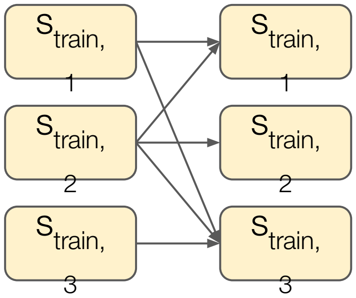
Problem Setting

We study three regimes depending on the relationship between S_{train} and S_{eval}:

Problem Setting

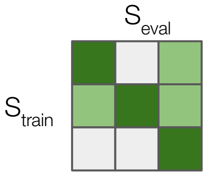
We study three regimes depending on the relationship between S_{train} and S_{eval}:

$S_{train} = S_{eval}$



Continual pre-training

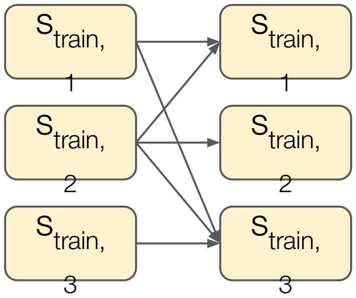
Adjacency matrix A of skills graph



Problem Setting

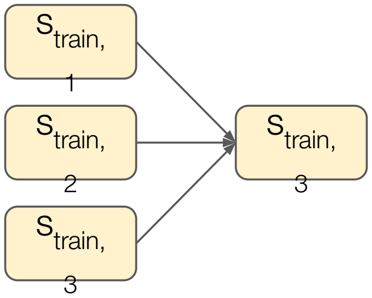
We study three regimes depending on the relationship between S_{train} and S_{eval}:

$S_{train} = S_{eval}$



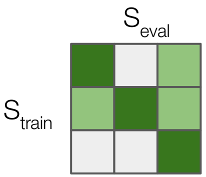
Continual pre-training

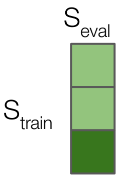
$S_{eval} \subset S_{train}$



Fine-tuning

Adjacency matrix A of skills graph

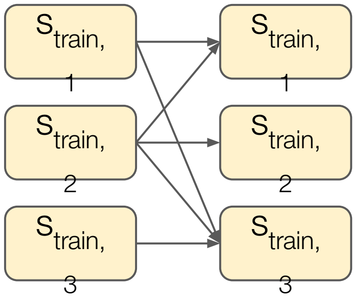




Problem Setting

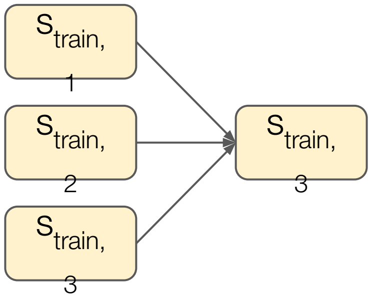
We study three regimes depending on the relationship between S_{train} and S_{eval}:

$S_{train} = S_{eval}$



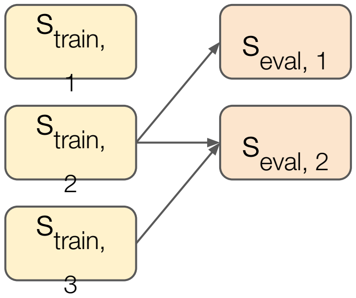
Continual pre-training

$S_{eval} \subset S_{train}$



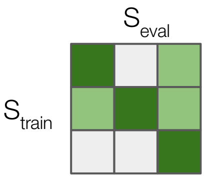
Fine-tuning

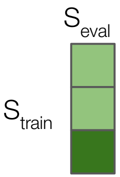
$S_{train} \cap S_{eval} = \varnothing$

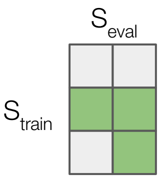


Out-of-domain

Adjacency matrix A of skills graph



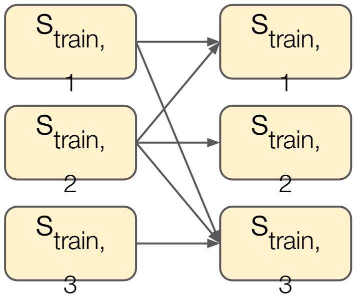




Problem Setting

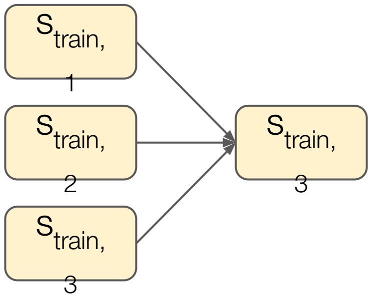
We study three regimes depending on the relationship between S_{train} and S_{eval}:

$S_{train} = S_{eval}$



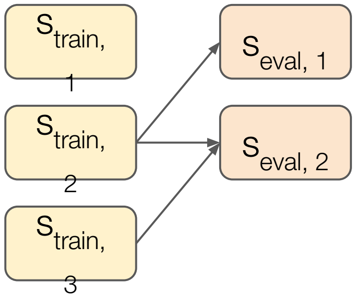
Continual pre-training

$S_{eval} \subset S_{train}$



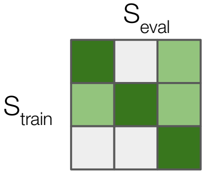
Fine-tuning

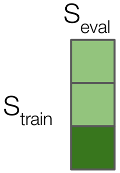
$S_{train} \cap S_{eval} = \varnothing$

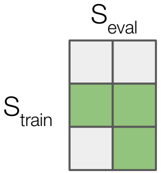


Out-of-domain

Adjacency matrix A of skills graph







Our general approach:
1. Learn A
2. Sample according to A

Skills Graph Learning

Can follow the definition:

- For each s_i, s_j, see if validation loss on s_j is lower when trained on s_i and s_j than when trained on just s_j for the same amount of steps, H
    - Set edge weight proportional to change in loss
    - Runtime: O(Hkm)

Skills Graph Learning

Can follow the definition:

- For each s_i, s_j, see if validation loss on s_j is lower when trained on s_i and s_j than when trained on just s_j for the same amount of steps, H
    - Set edge weight proportional to change in loss
    - Runtime: O(Hkm)

We can also make this cheaper!

- Linear approximation: for each s_i, s_j, see if validation loss on s_j decreases when trained on s_i
    - Set edge weight proportional to change in loss
    - Runtime: O(Hk)
- Reduce number of steps H
- Learn skills graph using smaller model

Sampling according to Skills Graph

Def: **skill-stratified sampling** involves sampling uniform from all skills that are prerequisite for the evaluation skill set.

- $S' = \{s_{train,i} \text{ if } \exists\ j \text{ s.t. } (s_{train,i}, s_{eval,\ j}) \in E\}$
- $Pr(s_{train,\ i}) = 1/|S| \text{ if } s_{train,i} \in S'$

Drawbacks: does not exploit skills graph dynamically - even after a skill is learned, we do not adjust skill proportions

Skill-It optimization problem

Formulate data selection as online optimization problem:

- T rounds: at each round t we sample n/T samples from X_{train} according to $p_t \in \Delta^{k-1}$ over the training skill set

Skill-It optimization problem

Formulate data selection as online optimization problem:

- T rounds: at each round t we sample n/T samples from X_{train} according to $p_t \in \Delta^{k-1}$ over the training skill set
- Let f_t be model at round t, with learning dynamics expressed as $f_t = \Phi(f_{t-1}, p_{t-1})$

Skill-It optimization problem

Formulate data selection as online optimization problem:

- T rounds: at each round t we sample n/T samples from X_{train} according to $p_t \in \Delta^{k-1}$ over the training skill set
- Let f_t be model at round t, with learning dynamics expressed as $f_t = \Phi(f_{t-1}, p_{t-1})$
- Let $L_{eval,\,j}(f_t)$ be the validation loss of f_t on $s_{eval,\,j}$.

Skill-It optimization problem

Formulate data selection as online optimization problem:

- T rounds: at each round t we sample n/T samples from X_{train} according to $p_t \in \Delta^{k-1}$ over the training skill set
- Let f_t be model at round t, with learning dynamics expressed as $f_t = \Phi(f_{t-1}, p_{t-1})$
- Let $L_{eval,\, j}(f_t)$ be the validation loss of f_t on $s_{eval,\, j}$.
- Objective: minimize average validation loss on evaluation skills at time T:

$$\underset{p_1,\ldots,p_T \in \Delta^{k-1}}{\text{minimize}} \quad \frac{1}{m}\sum_{j=1}^{m} L_{\text{eval},j}(f_T)$$

$$\text{s.t. } f_t = \Phi(f_{t-1}, p_{t-1}) \ \forall t \in 1, \ldots, T$$

Skill-It optimization problem

- Hard to solve without more constraints/information on Φ
- We think about learning dynamics in terms of how loss of model f_t depends on loss of f_{t-1} and proportions p_{t-1}

Skill-It optimization problem

- Hard to solve without more constraints/information on Φ
- We think about learning dynamics in terms of how loss of model f_t depends on loss of f_{t-1} and proportions p_{t-1}
- Simple idea (assume $S_{train} = S_{eval}$ for now):

$$L_{\text{eval},j}(f_t) = L_{\text{eval},j}(\Phi(f_{t-1}, p_{t-1})) := L_{\text{eval},j}(f_{t-1})(1 - \alpha p_{t-1}^j)$$

- Loss on j at time t is loss on j at time $t-1$ decreased by a factor proportional to p_{t-1}^j.
  - More training data from $X_{train,\, j}$ = loss goes down more
- Does not account for skills graph

Skill-It optimization problem

$$L_{\text{eval},j}(f_t) = L_{\text{eval},j}(\Phi(f_{t-1}, p_{t-1})) := L_{\text{eval},j}(f_{t-1})(1 - \alpha p_{t-1}^j)$$

Skill-It optimization problem

$$L_{\text{eval},j}(f_t) = L_{\text{eval},j}(\Phi(f_{t-1}, p_{t-1})) := L_{\text{eval},j}(f_{t-1})(1 - \boxed{A_{:,j}^\top p_{t-1})}$$

- More training data from $X_{\text{train, j}}$ or **any of *j*'s prerequisite skills** = loss on *j* goes down more

Skill-It optimization problem

$$\underset{p_1,\ldots,p_T \in \Delta^{k-1}}{\text{minimize}} \quad \frac{1}{m}\sum_{j=1}^{m} L_{\text{eval},j}(f_T)$$

$$\text{s.t. } L_{\text{eval},j}(f_t) = L_{\text{eval},j}(f_{t-1})(1 - A_{:,j}^\top p_{t-1})$$

$$\forall t \in [T], j \in [m]$$

Skill-It optimization problem

Derive Skill-It's update rule using online mirror descent:

- Regularize with negative entropy (similar to multiplicative weights)
- Proportion (unnormalized) for skill $s_{train,\, i}$ at time $t+1$:

$$p_{t+1}^{i} = p_{t}^{i} \exp\left(\eta \sum_{j=1}^{m} A_{ij} L_{\text{eval},j}(f_t)\right)$$

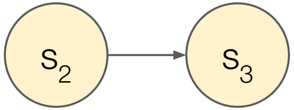
- $\eta > 0$ learning rate
- Intuitively: adjust weight on skill $s_{train,\, i}$ based on the losses of skills that $s_{train,\, i}$ influences
    - Key assumption: more data = validation loss goes down

Skill-It optimization problem

Toy example:

- $k = 3$ skills, $S_{train} = S_{eval}$
- Eta = 0.2
- Skills graph adjacency matrix:

	S_1	S_2	S_3
S_1	1	0	0
S_2	0	1	0.5
S_3	0	0	1

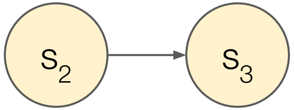


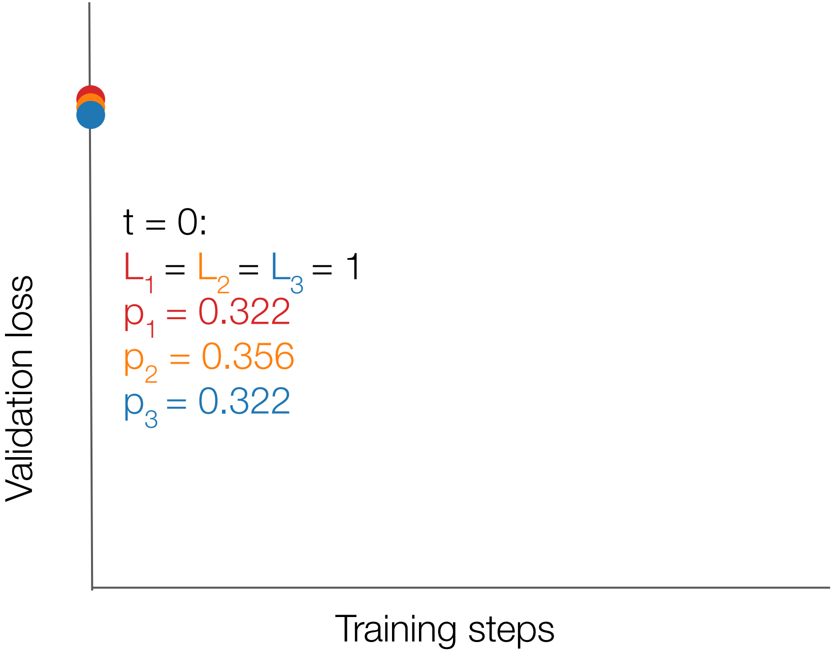
Skill-It optimization problem

Toy example:

- $k = 3$ skills, $S_{train} = S_{eval}$
- Eta = 0.2
- Skills graph adjacency matrix:

	S_1	S_2	S_3
S_1	1	0	0
S_2	0	1	0.5
S_3	0	0	1



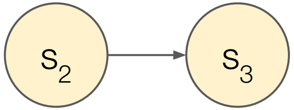


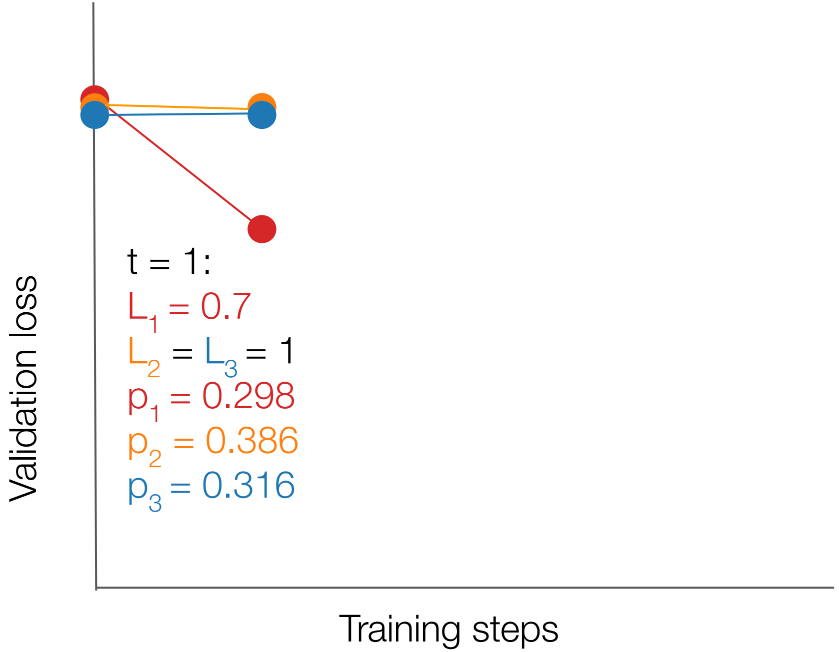
Skill-It optimization problem

Toy example:

- $k = 3$ skills, $S_{train} = S_{eval}$
- Eta = 0.2
- Skills graph adjacency matrix:

	S_1	S_2	S_3
S_1	1	0	0
S_2	0	1	0.5
S_3	0	0	1



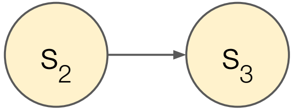


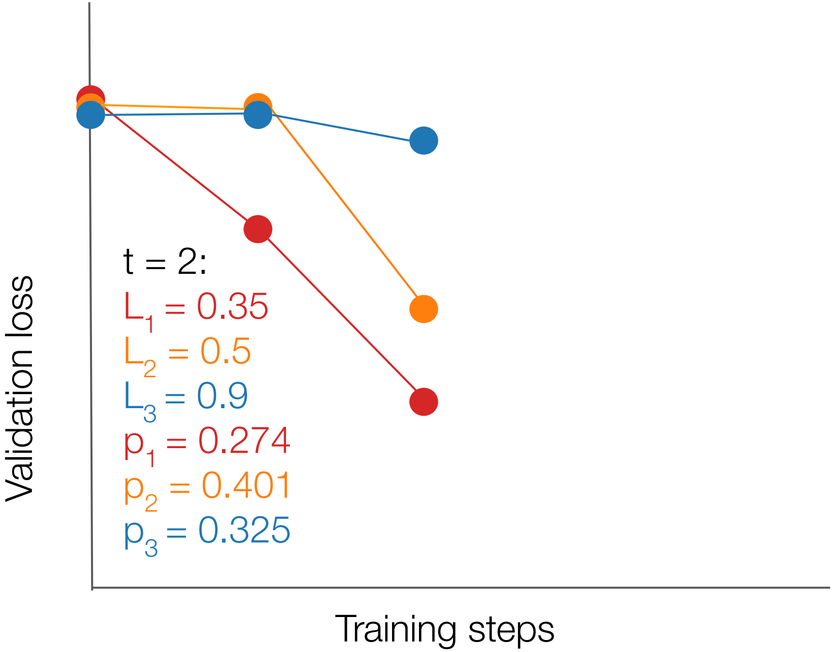
Skill-It optimization problem

Toy example:

- $k = 3$ skills, $S_{train} = S_{eval}$
- Eta = 0.2
- Skills graph adjacency matrix:

	S_1	S_2	S_3
S_1	1	0	0
S_2	0	1	0.5
S_3	0	0	1



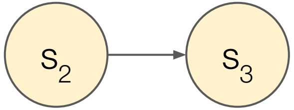


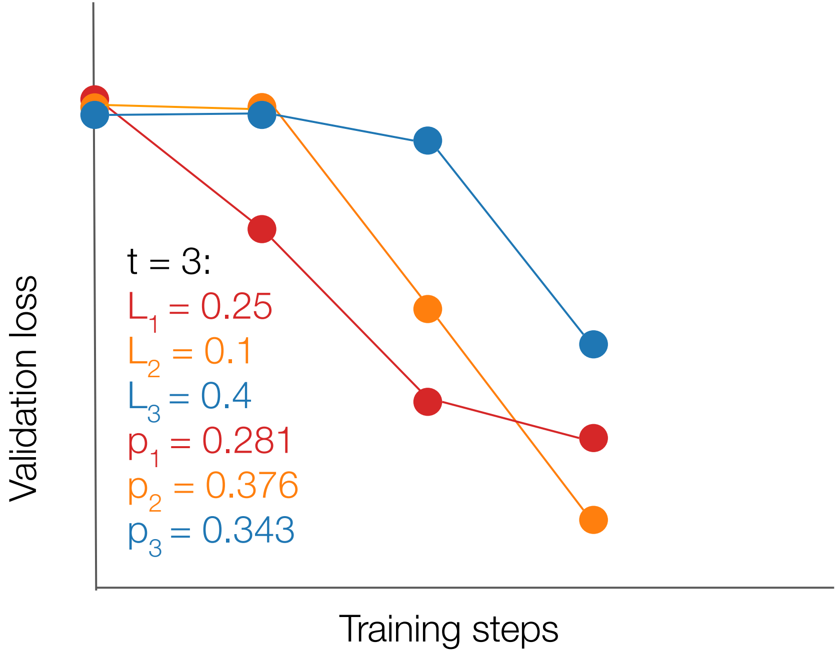
Skill-It optimization problem

Toy example:

- $k = 3$ skills, $S_{train} = S_{eval}$
- Eta = 0.2
- Skills graph adjacency matrix:

	S_1	S_2	S_3
S_1	1	0	0
S_2	0	1	0.5
S_3	0	0	1



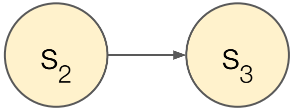


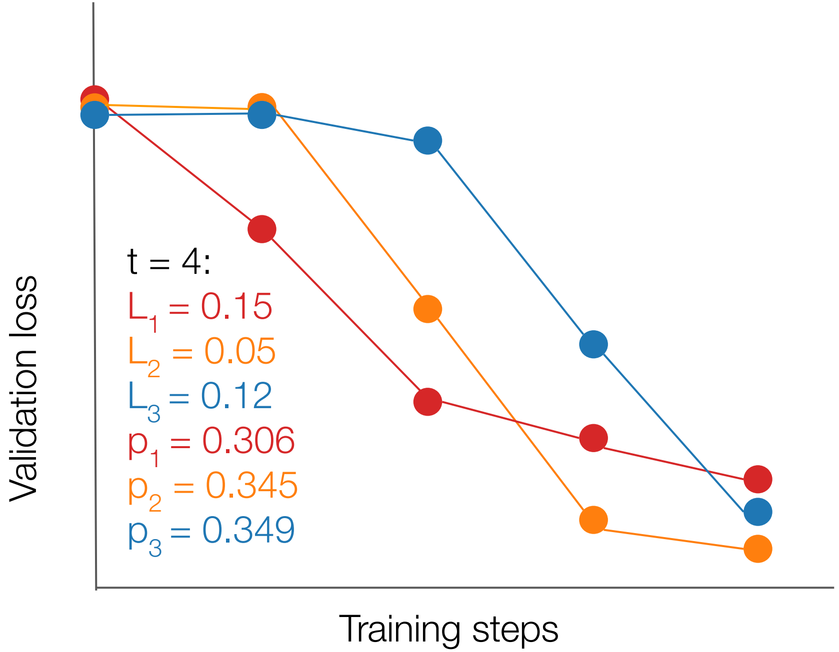
Skill-It optimization problem

Toy example:

- $k = 3$ skills, $S_{train} = S_{eval}$
- Eta = 0.2
- Skills graph adjacency matrix:

	S_1	S_2	S_3
S_1	1	0	0
S_2	0	1	0.5
S_3	0	0	1



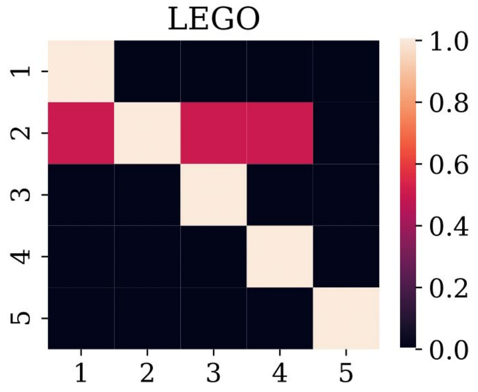


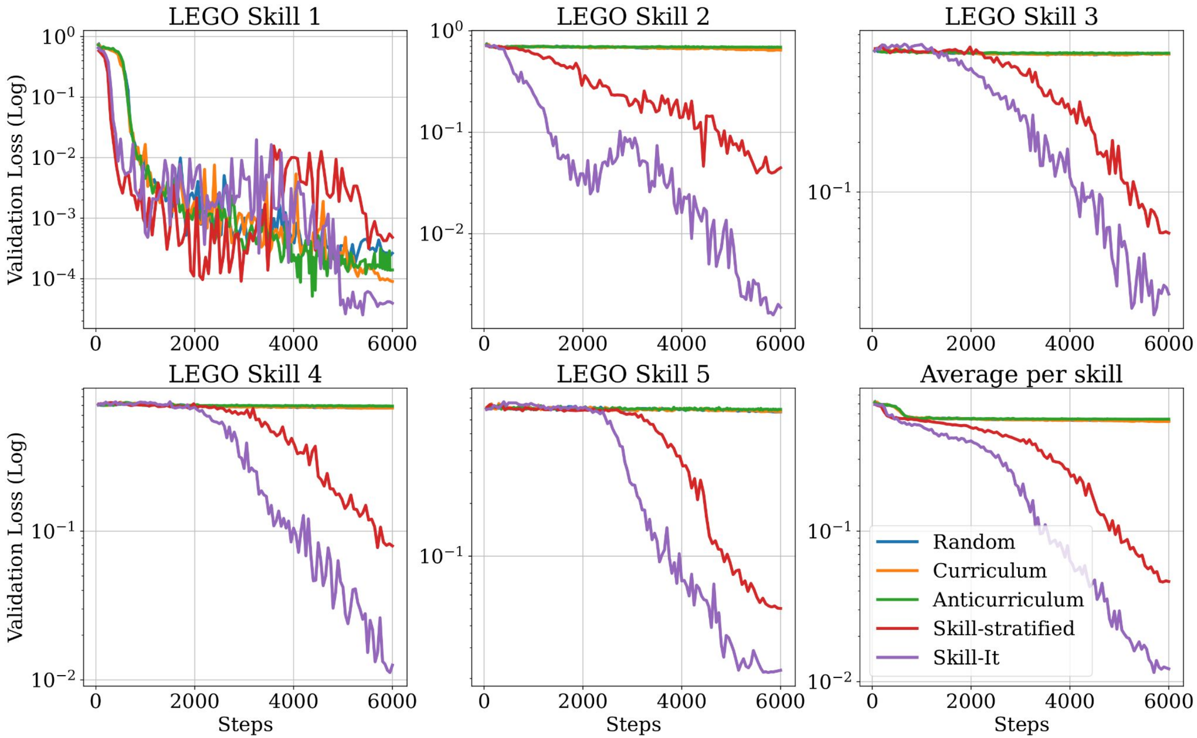
Results

Experiments Overview

- Evaluate Skill-It and skill-stratified sampling in three settings: continual pre-training, fine-tuning, out-of-domain
- Baselines:
  - Random sampling over training corpus or target skill (for fine-tuning)
  - For continual pre-training: curriculum learning at instance level and skill level
- Model: GPT-Neo-125M
  - Experiments where we learn skills graph on 125M and use it on GPT-Neo-1.3B
  - 3B parameter model with together.ai
- Datasets:
  - LEGO
  - Addition
  - Natural Instructions (various subsets)
  - RedPajama (together)

LEGO continual pre-training results





LEGO pre-training results

	Skill 1	Skill 2	Skill 3	Skill 4	Skill 5	Average
Random	$100.0_{\pm 0.0}$	$54.2_{\pm 5.9}$	$58.0_{\pm 3.1}$	$48.0_{\pm 6.3}$	$54.4_{\pm 7.3}$	$62.9_{\pm 3.5}$
Curriculum	$100.0_{\pm 0.0}$	$60.0_{\pm 10.6}$	$55.2_{\pm 5.8}$	$51.2_{\pm 6.3}$	$51.8_{\pm 6.1}$	$63.6_{\pm 3.6}$
Anticurriculum	$100.0_{\pm 0.0}$	$53.4_{\pm 2.3}$	$49.0_{\pm 4.8}$	$48.2_{\pm 6.4}$	$56.0_{\pm 5.7}$	$61.3_{\pm 2.2}$
Skill-stratified	$100.0_{\pm 0.0}$	$98.2_{\pm 1.8}$	$98.2_{\pm 1.3}$	$97.8_{\pm 1.6}$	$98.2_{\pm 1.3}$	$98.5_{\pm 0.9}$
Skill-curriculum	$100.0_{\pm 0.0}$	$75.2_{\pm 30.1}$	$52.2_{\pm 3.7}$	$51.0_{\pm 4.6}$	$54.4_{\pm 3.1}$	$66.6_{\pm 7.7}$
Skill-anticurriculum	$100.0_{\pm 0.0}$	$90.2_{\pm 8.1}$	$88.2_{\pm 8.3}$	$73.2_{\pm 12.2}$	$62.4_{\pm 9.4}$	$82.8_{\pm 4.9}$
SKILL-IT	$100.0_{\pm 0.0}$	$99.2_{\pm 0.8}$	$99.0_{\pm 1.0}$	$99.4_{\pm 0.5}$	$99.6_{\pm 0.5}$	$99.4_{\pm 0.2}$

Table 5: Results on accuracy per skill (binary classification) for LEGO pre-training experiment, averaged over 5 random seeds.

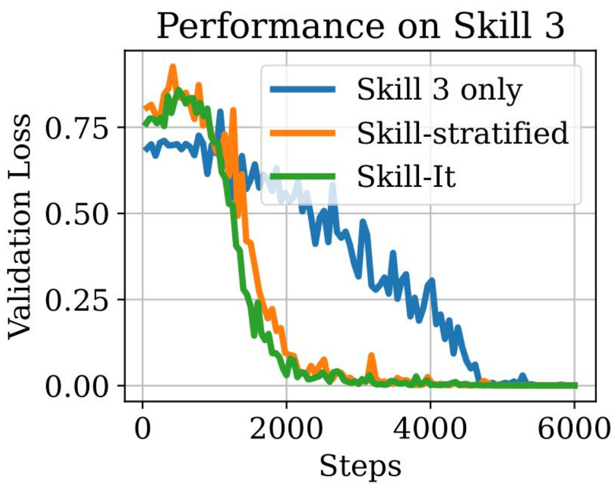
Natural instructions continual pre-training results

- Subset of 23 task categories (> 1M input/output pairs to select from)

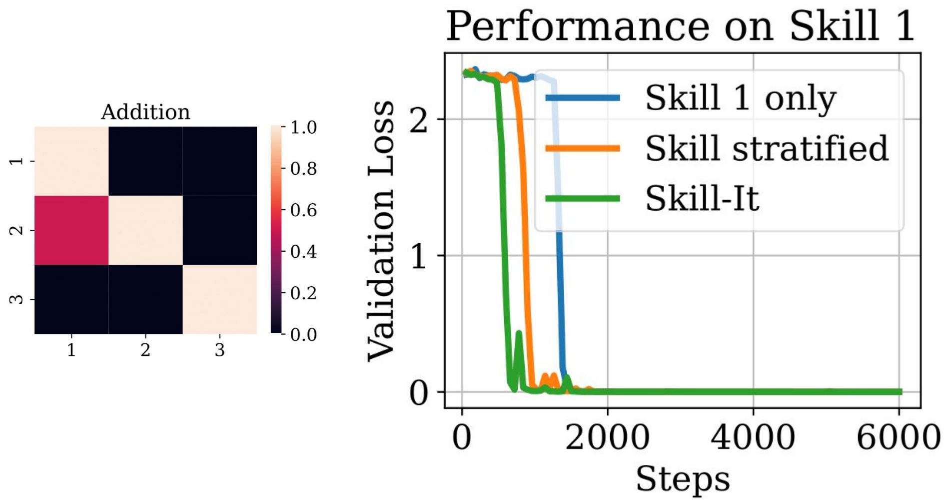
Method	Random	Curriculum	Anticurriculum	Skill curriculum	Skill anticuirrulum	Skill-stratified	Skill-it
Average val loss per skill	2.173 ± 0.028	2.307 ± 0.025	2.366 ± 0.026	2.304 ± 0.031	2.317 ± 0.052	2.115 ± 0.027	2.103 ± 0.032

LEGO and Addition fine-tuning results

LEGO: $S_{eval} = s_3$

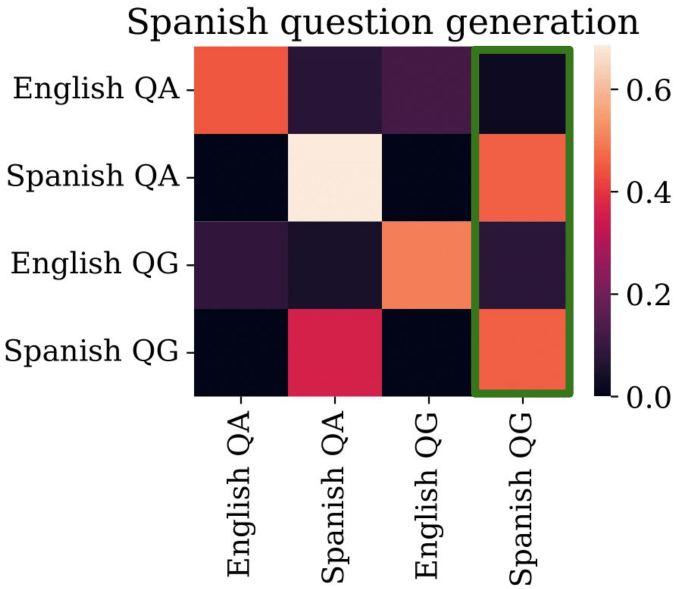


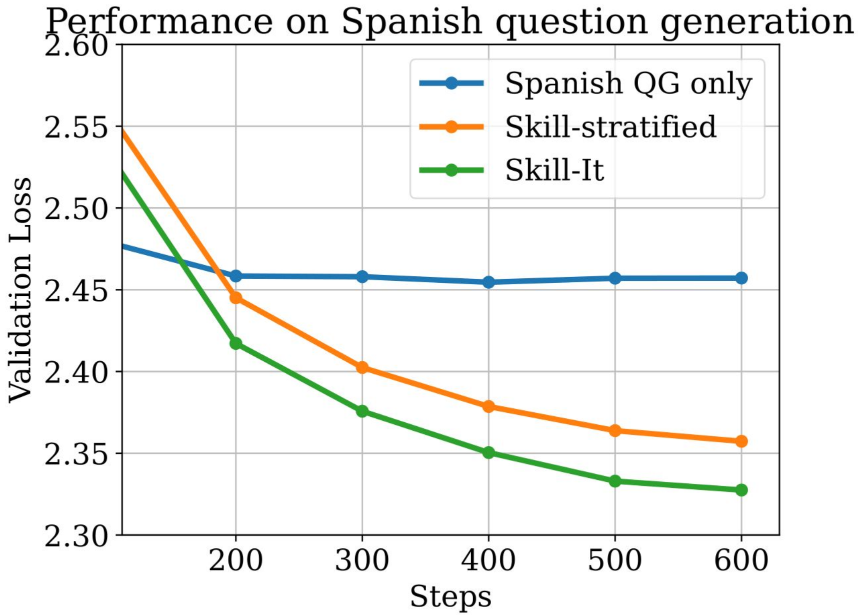
Addition: $S_{eval} = s_1$



Natural Instructions fine-tuning results

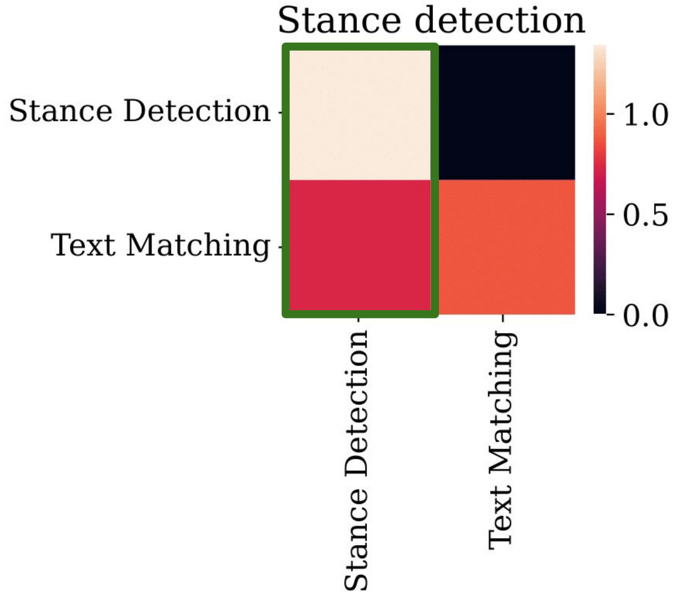
Spanish Question Generation

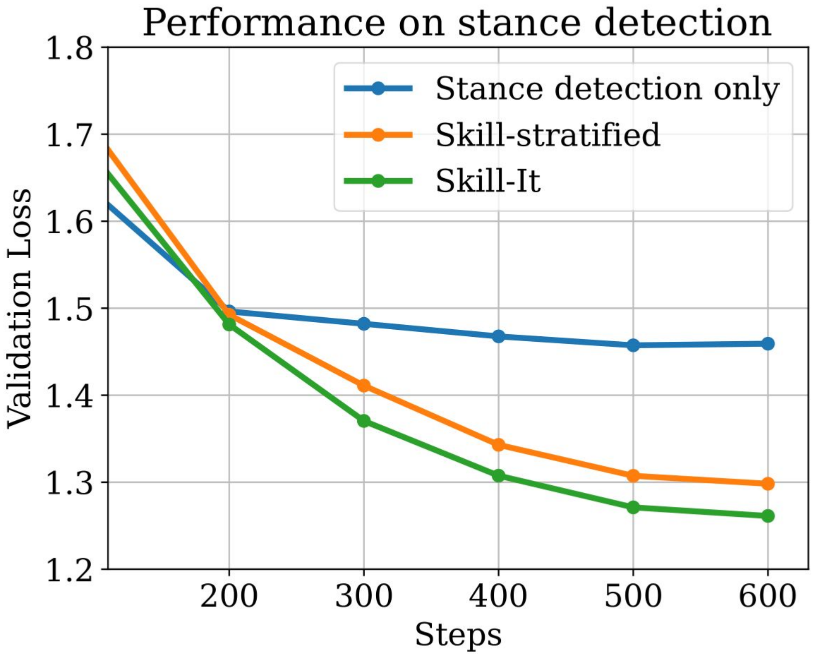




Natural Instructions fine-tuning results

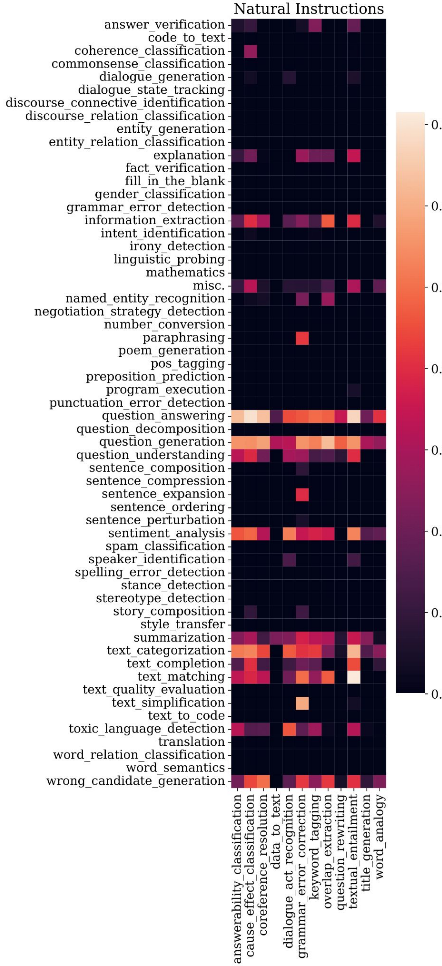
Stance Detection





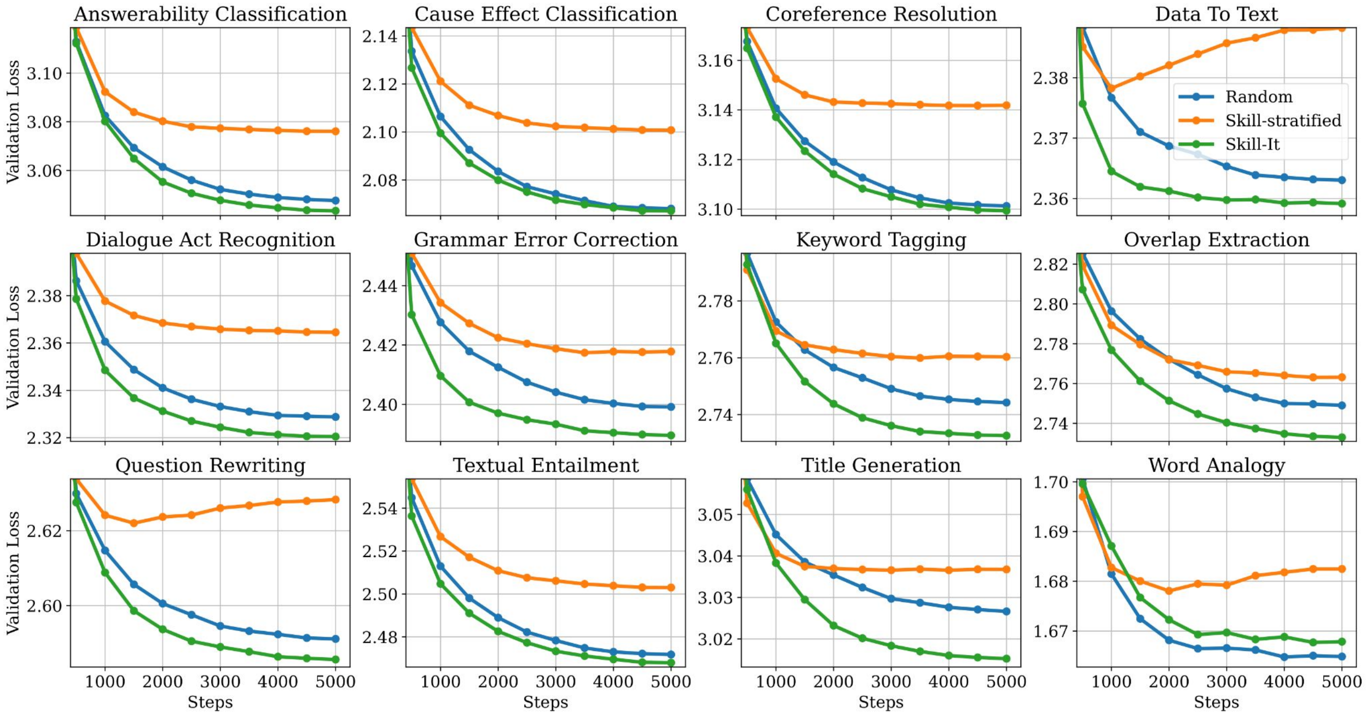
Natural instructions out-of-domain results

- Use 59 task categories from training split as training skill set
- Use 12 task categories from testing split as evaluation skill set



Natural instructions out-of-domain results

Skill-It outperforms baselines on 11/12 task categories



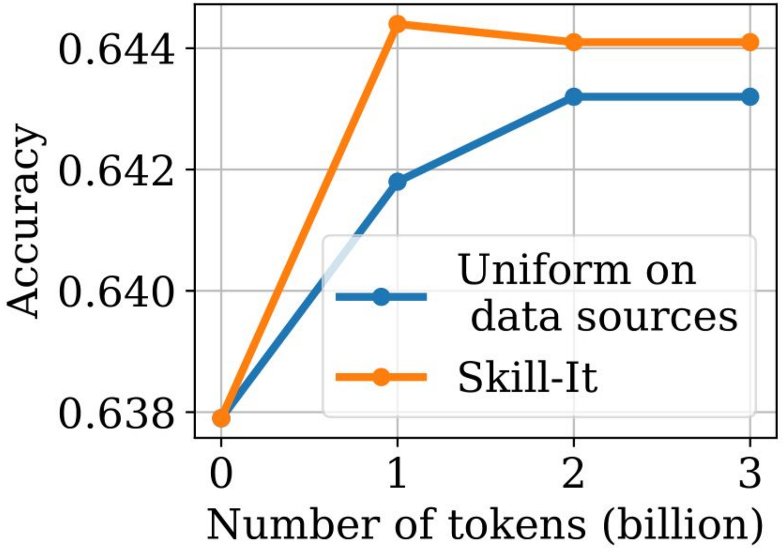
Large-scale case study: RedPajama[1]

- Training skill set: RedPajama's data sources (ArXiv, Books, C4, CommonCrawl, GitHub, StackExchange, Wikipedia)
- Evaluation skill set: Language Model Evaluation Harness[2] (ARC Challenge, Bool Q, Copa, HellaSwag, LAMBADA, PIQA, Winogrande)
- Continually pre-train a 3B parameter model already trained on 1T tokens for 3B additional tokens
- Set $T = 1$ (one round/static p_t)

[1] Together, 2023.
[2] Gao et. al. A framework for few-shot language model evaluation, 2021.

Large-scale case study: RedPajama

RedPajama source	SKILL-IT mixture
ArXiv	0.1370
Books	0.0437
C4	0.4195
CommonCrawl	0.0732
GitHub	0.189
StackExchange	0.0892
Wikipedia	0.0484

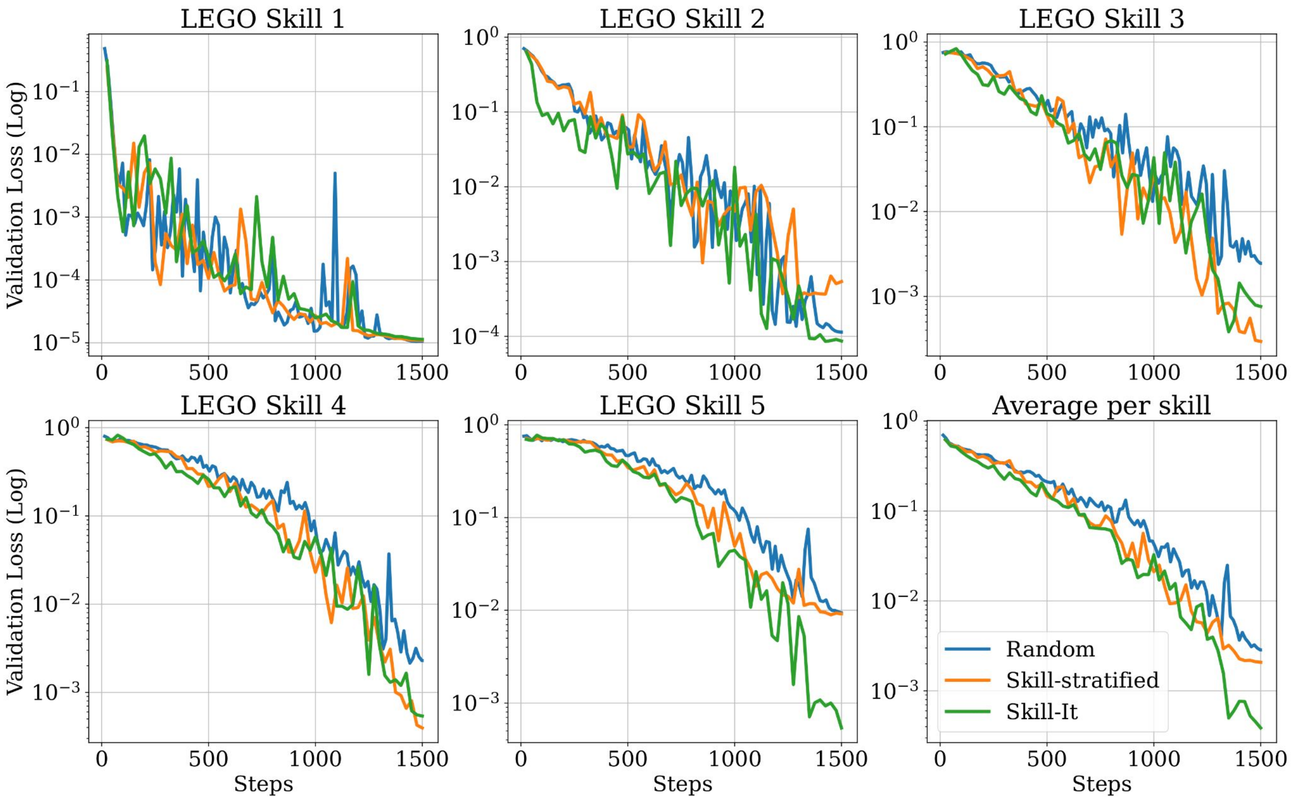


Transfer experiments

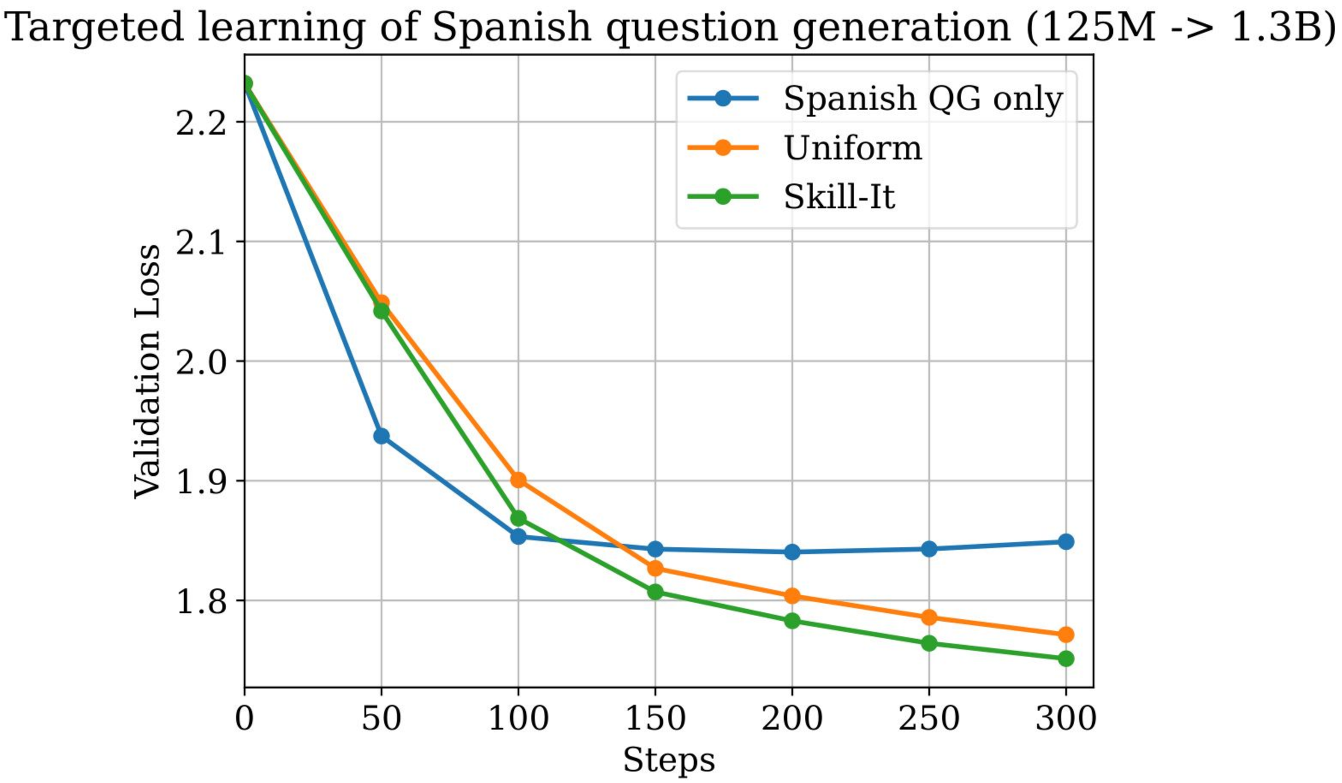
Can we learn the skills graph on a smaller model and use it on a larger model?

- Smaller model: GPT-Neo-125M
- Larger model: GPT-Neo-1.3B

Transfer experiments: LEGO continual pre-training

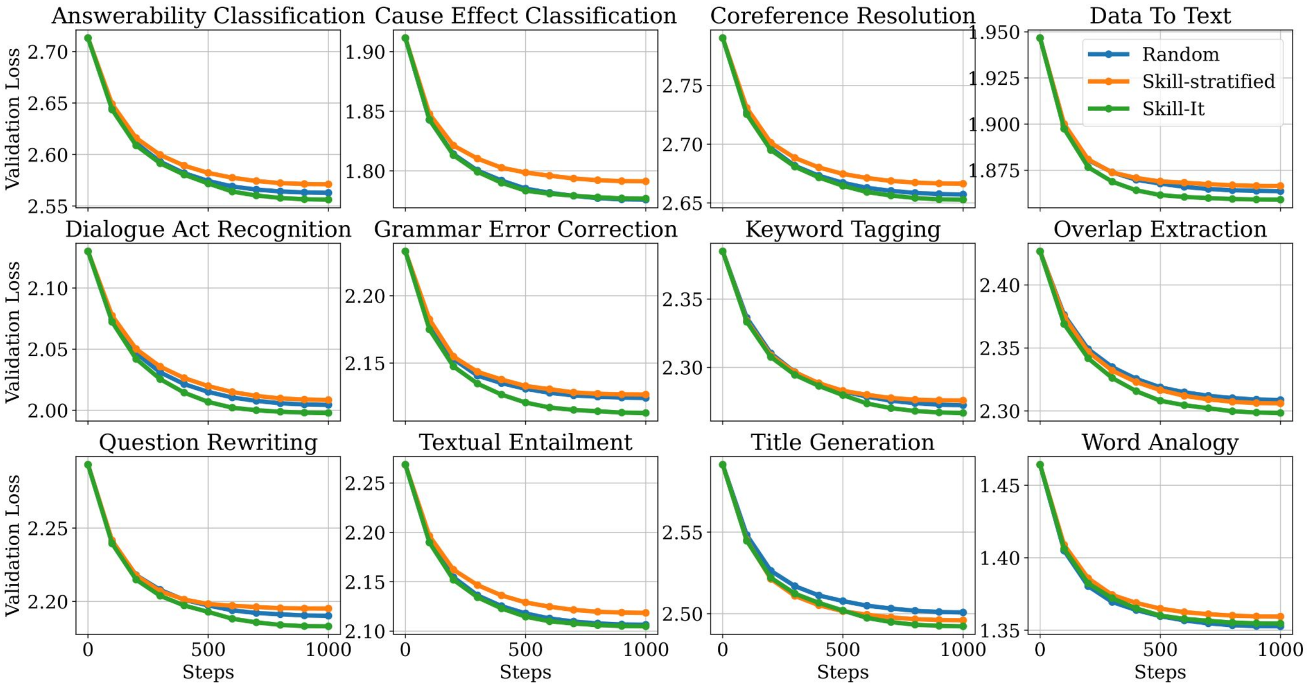


Transfer experiments: Natural Instructions fine-tuning



Transfer experiments: Natural Instructions out-of-domain

Skill-It outperforms baselines on 10/12 task categories.



Discussion

Summary

- Given a fixed budget, we want to know what data to train on in order to induce various capabilities in an LM
- We introduce a skills-based framework for understanding how LMs learn and for selecting training data
    - There exist ordered skill sets in the data that the model learns most efficiently in some particular order
    - We propose skill-It, an online algorithm for sampling from skills that uses the skills graph

Discussion

- The definition of skills and skills graph: intuition is not always correct!
- "Are skills just [tasks/data sources/etc.]?"
  - Not always. Skill are a **model-centric** concept
  - Bad set of candidate skills → empty or complete skills graph, Skill-It won't offer much gain in performance
  - **Discovering ordered skill set from data is hard** (see next slide).

Discussion

- The definition of skills and skills graph: intuition is not always correct!
- “Are skills just [tasks/data sources/etc.]?”
  - Not always. Skill are a **model-centric** concept
  - Bad set of candidate skills → empty or complete skills graph, Skill-It won’t offer much gain in performance
  - **Discovering ordered skill set from data is hard** (see next slide).
- “Can we use something simpler to get the skills graph?”
  - Edges are not always obvious (for example, LEGO)
    - Why does LEGO skill 2 help skill 3 and 4 while skill 3 does not help skill 4?
  - Tried using distances in embedding space/EMD, extremely noisy

Discussion

- The definition of skills and skills graph: intuition is not always correct!
- “Are skills just [tasks/data sources/etc.]?”
    - Not always. Skill are a **model-centric** concept
    - Bad set of candidate skills → empty or complete skills graph, Skill-It won’t offer much gain in performance
    - **Discovering ordered skill set from data is hard** (see next slide).
- “Can we use something simpler to get the skills graph?”
    - Edges are not always obvious (for example, LEGO)
        - Why does LEGO skill 2 help skill 3 and 4 while skill 3 does not help skill 4?
    - Tried using distances in embedding space/EMD, extremely noisy
- How to improve the Skill-It algorithm
    - Graph is *dynamic* and needs to be updated - how much can we do on the fly?

Future work: skills discovery

- Can we discover skills by clustering?
- Task: recover LEGO skills by clustering:
  - Embeddings
  - Gradients - in progress
  - Validation losses across multiple training runs
    - Points that have the same shifts/behaviors in validation loss as a function of training data tend to belong to the same skill

Cluster method	Accuracy
Pretrained embedding of last token	24.8 ± 0.5
Pretrained embedding of average token	25.2 ± 1.1
Trained model embedding of last token	38.4 ± 0.8
Sentence-BERT embedding	23.9 ± 0.7
Losses over multiple runs	$\mathbf{61.0 \pm 1.6}$

Thank you! Questions?

Contact: mfchen@stanford.edu

ArXiv: https://arxiv.org/abs/2307.14430