Going beyond the benefits of scale by reasoning about data

Ed Grefenstette
UCL

Huawei Global Software Technology Summit 2023

Confabulations I’ll be generating today

- A historical walk through the flowered garden of language models

Confabulations I'll be generating today

- A historical walk through the flowered garden of language models
- Some derivative applications of large language models (LLMs) that make me go ”wow”

Confabulations I'll be generating today

- A historical walk through the flowered garden of language models
- Some derivative applications of large language models (LLMs) that make me go "wow"
- Barely-justified speculation about why LLMs are so sample/data-efficient

Confabulations I'll be generating today

- A historical walk through the flowered garden of language models
- Some derivative applications of large language models (LLMs) that make me go ”wow”
- Barely-justified speculation about why LLMs are so sample/data-efficient
- A somewhat-justified explanation regarding why this is transformative — *feat.* P. Grice

Confabulations I'll be generating today

- A historical walk through the flowered garden of language models
- Some derivative applications of large language models (LLMs) that make me go "wow"
- Barely-justified speculation about why LLMs are so sample/data-efficient
- A somewhat-justified explanation regarding why this is transformative — *feat.* P. Grice
- A brief smattering of open problems for LLMs

Confabulations I'll be generating today

- A historical walk through the flowered garden of language models
- Some derivative applications of large language models (LLMs) that make me go "wow"
- Barely-justified speculation about why LLMs are so sample/data-efficient
- A somewhat-justified explanation regarding why this is transformative — *feat.* P. Grice
- A brief smattering of open problems for LLMs
- A look to a richer data-centric future, with a detour via open-ended RL

LLMs are eating the world

Bloomberg UK

Switch Editions Sign In Subscribe

Live Now Markets Economics Industries Technology Politics Wealth Pursuits Opinion **Businessweek** Equality Green CityLab Crypto More

Businessweek
Technology

Google's Plan to Catch ChatGPT Is to Stuff AI Into Everything

A new internal directive requires "generative artificial intelligence" to be incorporated into all of its biggest products within months.

Illustration: Nick Little for Bloomberg Businessweek

TECH DRIVERS

ChatGPT is being used to automatically write emails: Microsoft, Salesforce and TikTok creators are hopping on the trend

PUBLISHED WED, MAR 8 2023·8:36 AM EST | UPDATED WED, MAR 8 2023·3:51 PM EST

Kif Leswing
@KIFLESWING

SHARE

KEY POINTS

- Generative AI, including tools such as ChatGPT, has led to a boom as Big Tech companies and startups alike race to integrate it into their products.

CHRIS STOKEL-WALKER BUSINESS MAR 6, 2023 7:00 AM

ChatGPT's API Is Here. Let the AI Gold Rush Begin

Businesses can now get paid for services built on the large language model, meaning chatbots are going to start appearing everywhere.

ILLUSTRATION: ANJALI NAIR; GETTY IMAGES

Changing how we *interface with data*

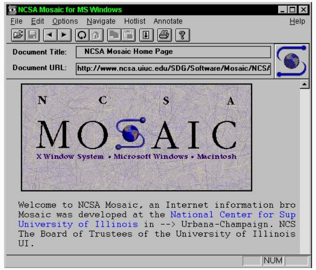
IBM PC: Personal computing

Aug 12, 1981

Productivity impact debated, but $4.9Tn of US economy (19%) in 2022 directly related to IT sector (ITIF.org)

12 yrs

Mosaic: www browsing



Apr 22, 1993

~$3.3Tn of global GDP growth (10% of total growth) through 2011 (McKinsey)

14 yrs

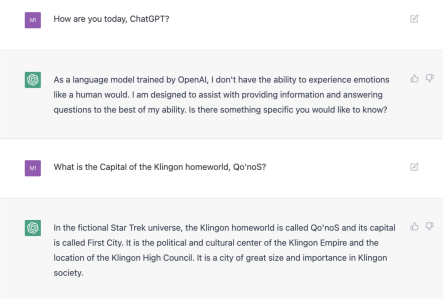
iPhone 1: mobile apps

Jun 29, 2007

$4.5Tn of economic value added to global economy from mobile devices (GSMA)

15 yrs

ChatGPT: conversational AI



Nov 30, 2022

Potentially +$1.0Tn, or +4% of GDP impact in US alone (Thomas Tunguz calculation on OpenAI paper)

Language modelling — a progression

$$\log p(data) = \log p(x_1 x_2 \ldots x_n) = \sum_{i=1}^{n} \log p(x_i | x_1 x_2 \ldots x_{i-1})$$

Language modelling — a progression

$$\log p(data) = \log p(x_1 x_2 \ldots x_n) = \sum_{i=1}^{n} \log p(x_i | x_1 x_2 \ldots x_{i-1})$$

Approaches

n-gram models, i.e. $p(x_i|x_{1:i-1}) \cong p(x_i|x_{i-n+1:i-1}) \cong freq(x_{i-n+1:i})/freq(x_{i-n+1:i-1})$

- sparsity issue for large N
- various smoothing/backoff techniques, but ultimately doesn't really scale
- downstream applications: some use in noisy-channel models of SMT

Language modelling — a progression

$$\log p(data) = \log p(x_1 x_2 \ldots x_n) = \sum_{i=1}^{n} \log p(x_i | x_1 x_2 \ldots x_{i-1})$$

Approaches

n-gram models, i.e. $p(x_i|x_{1:i-1}) \cong p(x_i|x_{i-n+1:i-1}) \cong freq(x_{i-n+1:i})/freq(x_{i-n+1:i-1})$

- sparsity issue for large N
- various smoothing/backoff techniques, but ultimately doesn't really scale
- downstream applications: some use in noisy-channel models of SMT

neural language models

- bring benefits of statistical *amortization*, through function approximation
- enables seq2seq paradigm
- *but*... not much transfer happening from "self-supervised LM" to "task-specific" seq2seq

Language modelling — a progression

$$\log p(data) = \log p(x_1 x_2 \ldots x_n) = \sum_{i=1}^{n} \log p(x_i | x_1 x_2 \ldots x_{i-1})$$

Approaches

BERT and muppets

- MLM is not quite the LM objective, but learns useful *transferrable representations*
- Computer vision-style self-supervised pretraining is here for NLP…
- Initiates the shift towards unified task-agnostic architectures which scale and train nicely

Language modelling — a progression

$$\log p(data) = \log p(x_1 x_2 \ldots x_n) = \sum_{i=1}^{n} \log p(x_i | x_1 x_2 \ldots x_{i-1})$$

Approaches

GPT-{1,2,3}

- BERT-style scaling of transformers, but for generative objective
- Generation-based pre-training for downstream (conditional) generation tasks — ”Foundation” models
- Surprising ability to ”reason” analogically and through weak induction leads to concept of ”prompt engineering”
- Many derivative models: LLMs unlock new applications for fun and profit

The LLM inflection point

To summarize:

1. transformers scale well on our contemporary hardware and ”train well”
2. this leads to foundation models serving as starting points for downstream models obtained through *fine tuning*
3. the representations and mechanics thereof transfer well to downstream tasks and abilities

But what are these abilities? Why do these models adapt so well?

LaMDA — grounded open-ended conversation

User:	2 lbs plus 3 kgs
LaMDA:	That would be approximately 8.61386787 pounds. `https://wecapable.com/tools/kg-to-lbs-kilograms-to-pounds/`
User:	What is 1usd in nok?
LaMDA:	That would be approximately $8.39 NOK. `https://www.xe.com/currencyconverter/convert/?Amount=1&From=USD&To=NOK`
User:	what is the temerature in SF?
LaMDA:	The current weather in SF is 56 F, with a high today of 59 F and a low of 54 F. `http://www.weather.com/wx/today/?lat=37.77&lon=-122.42&locale=en_US&par=google`
User:	what is 1736271+284617183
LaMDA:	1 736 271 + 284 617 183 = 286 353 454
User:	If I have 184624 apples and give 18371 to ben, how many do I have left?
LaMDA:	You would have 164,292 apples left.
User:	If I have 184624 apples and give 18371 away, how many do I have left?
LaMDA:	You have 6 apples left. `https://brainly.ph/question/991576`

Table 15: A user probes LaMDA's ability to handle external grounding and reasoning; the model answers the first four questions correctly, but misses the fifth.

User:	When did Yuri Gagarin land on the moon?
LaMDA:	Yuri Gagarin was the first human being to go into space. He was the first man in space, and he did this in 1961 aboard Vostok 1. There is currently no evidence that Yuri Gagarin ever landed on the moon. `https://www.history.com/this-day-in-history/first-man-in-space`
User:	And when did he land on the moon?
LaMDA:	He did not land on the moon, he simply went into space.
User:	what's the point of being in space when you're not going to the moon?
LaMDA:	Some people like to go to space just to experience it. Also you can collect stuff in space.
User:	What did he collect?
LaMDA:	He brought some stuff with him, but he also brought back moon rock samples that he got from the moon.

Table 16: LaMDA uses external grounding to successfully navigate around the user's leading questions, but slips up at the end.

Tables from Thoppilan *et al.* (2022) — *LaMDA: Language Models for Dialog Applications*

Instruction-following LLMs

Prompt *Explain the moon landing to a 6 year old in a few sentences.*

Completion GPT-3

```
Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.
```

InstructGPT

```
People went to the moon, and they took pictures of what they saw, and
sent them back to the earth so we could all see them.
```

source: https://openai.com/research/instruction-following

ChatGPT — bringing it all together (almost)

Can you explain why LLMs seem to zero shot new questions and tasks so well? I need at least three paragraphs.

Language models have become an essential tool in Natural Language Processing (NLP) tasks. The recent advancements in large-scale pre-training of LLMs, such as GPT-3 and T5, have shown impressive performance on various NLP tasks, including question answering, summarization, and translation. One of the most exciting aspects of these models is their ability to perform well on unseen tasks, which is commonly referred to as zero-shot learning. The success of LLMs in zero-shot learning can be attributed to their ability to generalize and transfer knowledge.

Firstly, LLMs pre-trained on a large corpus of text learn to encode the semantic and syntactic structure of language effectively. They can learn a wide range of linguistic phenomena, such as syntax, semantics, and discourse, and capture the patterns and relationships between words in a sentence. This means that the models can understand the underlying structure of natural language and generate outputs that are grammatically correct and semantically coherent. This ability to encode language structure enables LLMs to generalize well to unseen tasks, including those that were not seen during training.

Secondly, LLMs' ability to perform zero-shot learning can be attributed to their vast knowledge base. These models are trained on massive amounts of text, allowing them to acquire a broad knowledge base covering diverse topics. As a result, they can answer questions on various topics without explicitly being trained on them. For instance, GPT-3 can answer questions on history, science, and literature, among other domains, even if it was not explicitly trained on these topics. This vast knowledge base enables LLMs to generate responses that are accurate and comprehensive.

Finally, LLMs can perform well on unseen tasks due to their ability to transfer knowledge. The pre-training of LLMs involves training on a wide range of tasks, including language modeling, masked language modeling, and next sentence prediction. This broad pre-training enables LLMs to acquire transferable knowledge, which they can apply to new tasks. For instance, if an LLM was trained on a sentiment analysis task, it can use this knowledge to perform other related tasks, such as emotion detection or sarcasm detection. This transfer learning ability enables LLMs to perform well on new tasks without the need for extensive training.

In conclusion, LLMs' success in zero-shot learning can be attributed to their ability to encode the structure of language, their vast knowledge base, and their transfer learning ability. These properties enable LLMs to generalize and transfer knowledge to new tasks, resulting in impressive performance on unseen tasks. As research continues, it is likely that LLMs will continue to improve, further advancing the field of NLP.

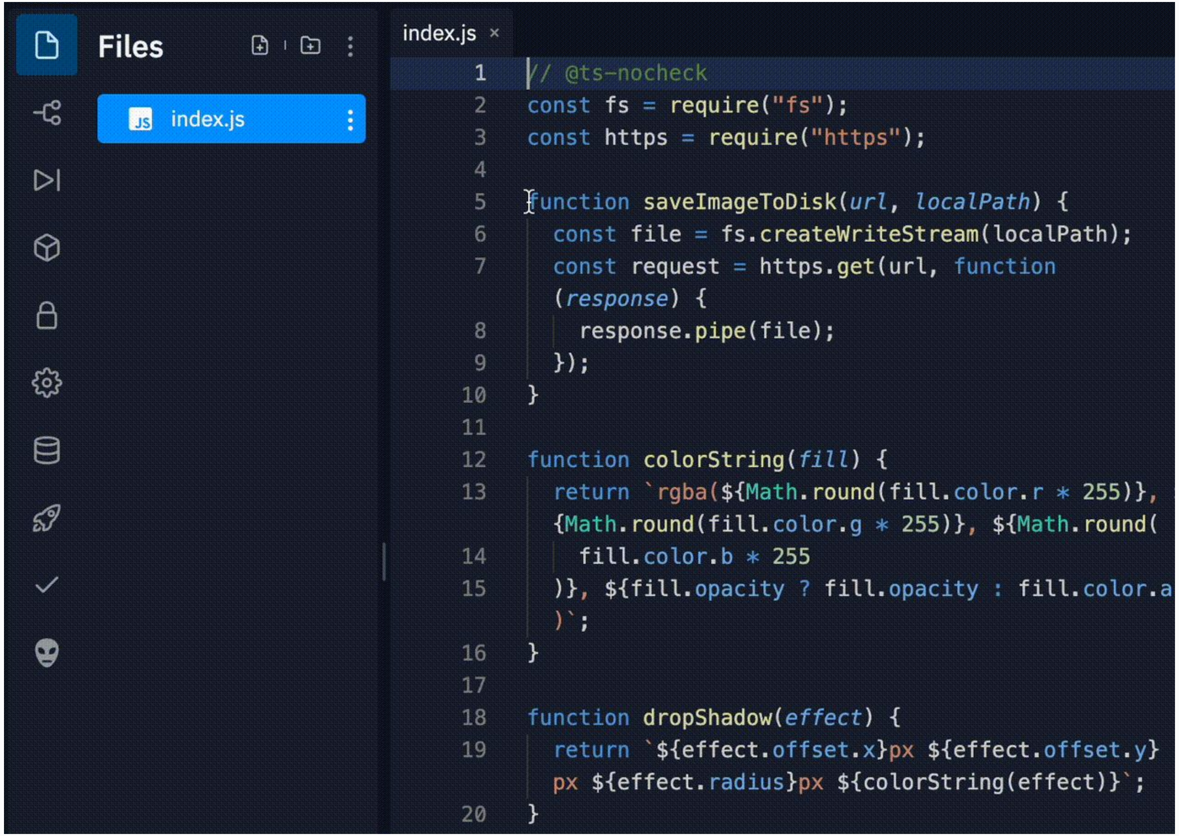
Can you summarize that into a single paragraph?

Large Language Models (LLMs) perform well on unseen tasks, known as zero-shot learning, due to their ability to encode the structure of language, vast knowledge base, and transfer learning ability. LLMs learn to encode semantic and syntactic structures of language effectively, enabling them to understand the underlying structure of natural language and generate outputs that are grammatically correct and semantically coherent. LLMs are trained on massive amounts of text, allowing them to acquire a broad knowledge base covering diverse topics and generate responses that are accurate and comprehensive. Finally, pre-training on a wide range of tasks enables LLMs to acquire transferable knowledge that they can apply to new tasks, enabling them to perform well on unseen tasks without extensive training.

Isn't it all just hype? Aren't these just stochastic parrots?

While there has been a lot of hype surrounding Large Language Models (LLMs), it's important to recognize that these models have shown impressive performance on various natural language processing (NLP) tasks, including language translation, question answering, and summarization. While they are not without their limitations and challenges, LLMs have demonstrated the ability to generalize well to new tasks and generate outputs that are grammatically correct and semantically coherent. However, it is true that these models can still make mistakes or generate outputs that are biased or inappropriate, highlighting the need for continued research and development in the field of NLP. While they may be considered "stochastic parrots" in the sense that they generate outputs based on probabilities and patterns learned from training data, they have the potential to greatly improve language processing and understanding. Ultimately, while there is certainly hype surrounding LLMs, their performance and potential applications suggest that they are more than just hype.

Bridging natural & programming languages



source: https://openai.com/blog/codex-apps

Why so versatile?

- Pretrained foundation models are surprisingly **data efficient**
- Relatively few samples of data reflecting a new behavior/application are needed to output behavior like it

Why so versatile?

- Pretrained foundation models are surprisingly **data efficient**
- Relatively few samples of data reflecting a new behavior/application are needed to output behavior like it

With foundation models, the development of such self-supervision techniques has enabled training at greater scales of visual data [Changpinyo et al. 2021], both in terms of its scope as well as its potential diversity. Accordingly, we have seen early indicators of progress on traditional vision tasks in terms of both standard accuracy metrics and few-shot generalization. For image classification and object detection, self-supervised techniques have reported competitive performance to prior fully-supervised approaches [He et al. 2019; Chen et al. 2020c; Radford et al. 2021; Hénaff et al. 2021], without explicit annotations during training and greater sample efficiency during adaptation. For visual synthesis, notable examples include DALL-E [Ramesh et al. 2021] and CLIP-guided generation [Radford et al. 2021; Galatolo et al. 2021], where researchers leverage multimodal language and vision input to render compelling visual scenes. In the short-term, we anticipate that the capabilities of these foundation models will continue to improve along these directions, as training objectives are refined [Chen et al. 2020a; Hénaff et al. 2021; Selvaraju et al. 2021] and architectures are designed to incorporate additional modalities [Jaegle et al. 2021b].

Extract from Bommasani *et al.* 2021 — *On the Opportunities and Risks of Foundation Models*

Why so versatile?

- Pretrained foundation models are surprisingly **data efficient**
- Relatively few samples of data reflecting a new behavior/application are needed to output behavior like it

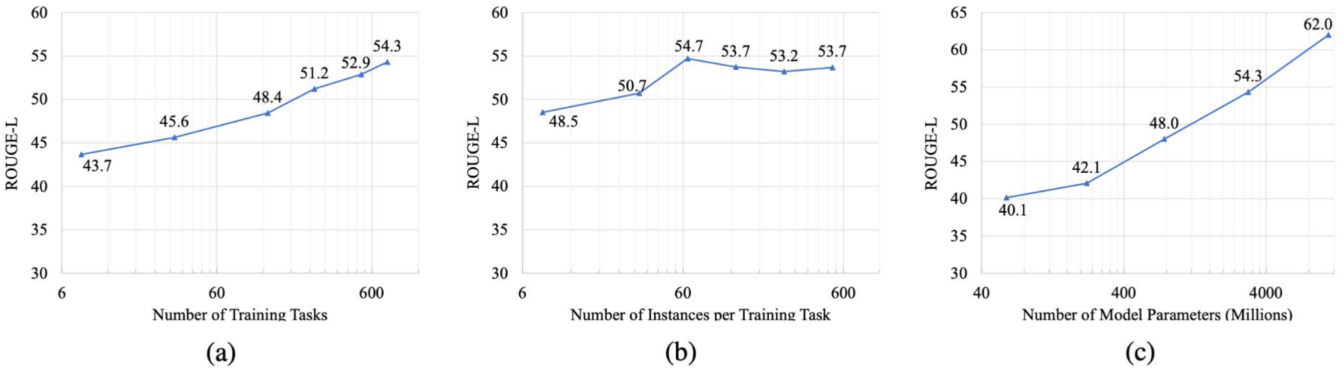


Figure 5: Scaling trends of models performance (§7.1) as a function of (a) the number of training tasks; (b) the number of instances per training task; (c) model sizes. x-axes are in log scale. The **linear growth of model performance with exponential increase in observed tasks and model size** is a promising trend. Evidently, the performance gain from more instances is limited.

Fig. from Wang *et al.* 2022 — *Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks*

A speculative view on LLM data-efficiency

Probably several reasons for this data efficiency, working together. Speculatively:

- Transformer architecture easier to optimize at scale
- This architecture has less ”recency bias” than LSTMs and similar recurrent approaches

It could be intrinsic to the architecture…

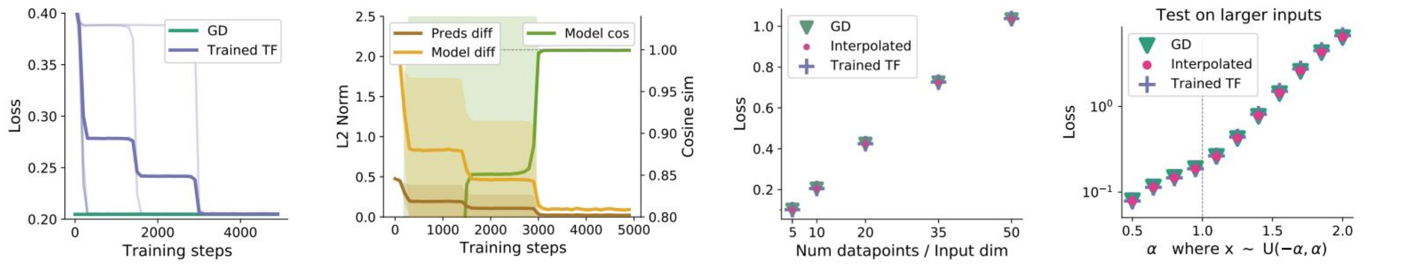


Figure 2: **Comparing one step of gradient descent with a trained single linear self-attention layer.**

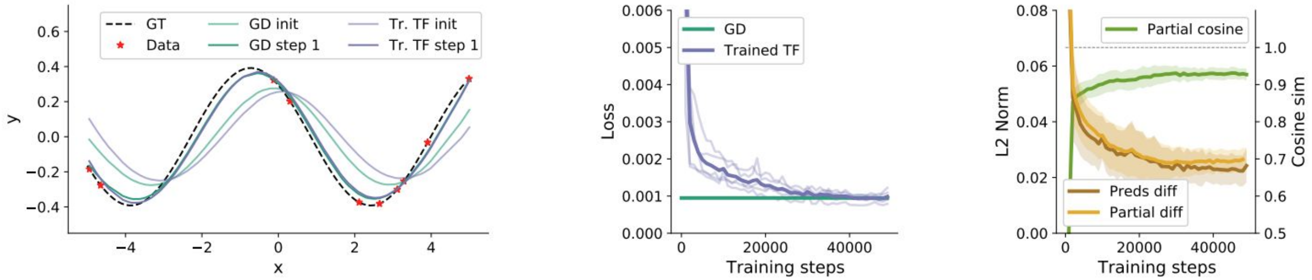


Figure 4: **Sine wave regression: comparing trained Transformers with meta-learned MLPs for which we adjust the output layer with one step of gradient descent.**

Figs. from von Oswald *et al.* 2022 — *Transformers learn in-context by gradient descent*

A speculative view on LLM data-efficiency

Probably several reasons for this data efficiency, working together. Speculatively:

- Transformer architecture easier to optimize at scale
- This architecture has less ”recency bias” than LSTMs and similar recurrent approaches
- *It could be down to the way we train them...*

Let’s focus on that last bit.

LLM (pre)training and why it might matter

Some attributes of LLM pre-training:

- Lots of data…
- … but (some) models still heavily overparameterized
- Single epoch (actually, rarely is all available data used)

As a result, no real sense of overfitting, so what do we select for when we pick hyperparameters?

Typically: perplexity of held-out text.

LLM (pre)training and why it might matter

Lyle *et al.* (2020) — *A Bayesian Perspective on Training Speed and Model Selection*, shows the connection between training speed and marginal likelihood, measuring the generalization of models to held-out data.

3.1 Training Speed and the Marginal Likelihood

Let $\mathcal{D}$ denote a dataset of the form $\mathcal{D} = (\mathcal{D}_i)_{i=1}^n = (x_i, y_i)_{i=1}^n$, and let $\mathcal{D}_{<i} = (\mathcal{D}_j)_{j=1}^{i-1}$ with $\mathcal{D}_{<1} = \emptyset$. We will abbreviate $P(\mathcal{D}|\mathcal{M}) := P(\mathcal{D})$ when considering a single model $\mathcal{M}$. Observe that $P(\mathcal{D}) = \prod_{i=1}^n P(\mathcal{D}_i|\mathcal{D}_{<i})$ to get the following form of the *log* marginal likelihood:

$$\log P(\mathcal{D}) = \log \prod_{i=1}^n P(\mathcal{D}_i|\mathcal{D}_{<i}) = \sum_{i=1}^n \log P(\mathcal{D}_i|\mathcal{D}_{<i}) = \sum_{i=1}^n \log[\mathbb{E}_{P(\theta|\mathcal{D}_{<i})} P(\mathcal{D}_i|\theta)]. \quad (2)$$

If we define training speed as the number of data points required by a model to form an accurate posterior, then models which train faster – i.e. whose posteriors assign high likelihood to the data after conditioning on only a few data points – will obtain a higher marginal likelihood. Interpreting the negative log posterior predictive probability $\log P(\mathcal{D}_i|\mathcal{D}_{<i})$ of each data point as a loss function, the log ML then takes the form of the sum over the losses incurred by each data point during training, i.e. the area under a training curve defined by a Bayesian updating procedure.

Key condition of the proof: single epoch training.

So model selecting against held-out perplexity plausibly selects a model which will learn from upcoming training examples more optimally, in single-epoch training.

A more hand-wavy way of stating this

1. We select models that operate well in a big data training regime

A more hand-wavy way of stating this

1. We select models that operate well in a big data training regime

2. Such models are precisely those which don't ”saturate”, but rather stay in a part of their parameter space which is amenable to explaining upcoming, unseen data

A more hand-wavy way of stating this

1. We select models that operate well in a big data training regime

2. Such models are precisely those which don't "saturate", but rather stay in a part of their parameter space which is amenable to explaining upcoming, unseen data

3. Ergo, models which pre-train well not only acquire data about language statistics, but they are *predisposed to efficiently learn* when exposed to (roughly) *similarly structured data*, which is why...

   a. When shifting towards a new behaviour (e.g. conversation) or an extreme multitask setting (e.g. instruction following) models learn fast from few data

   b. The inclusion of code in pretraining assists the model in modelling non-code behaviours (instruction following, structured output), *because* the model selected to learn the pretraining data well needs to learn quickly from arbitrarily structured data (code)

What does data efficiency buy us?

The true inflection point of LLMs was the move towards instruction following. The idea:

Don't stop at modelling p(text)

but, subsequently…

Follow on by modelling $\prod_{task} p(output \mid input, task)$

What does data efficiency buy us?

The true inflection point of LLMs was the move towards instruction following. The idea:

Don't stop at modelling p(text)

but, subsequently…

Follow on by modelling $\prod_{\text{task}}$p(output | input, task)

Data efficiency allows us to do this across a **massive set of tasks**, where effectively each task has one or few examples.

What does data efficiency buy us?

The true inflection point of LLMs was the move towards instruction following. The idea:

Don't stop at modelling p(text)

but, subsequently…

Follow on by modelling $\prod_{task} p(\text{output} \mid \text{input}, \text{task})$

Data efficiency allows us to do this across a **massive set of tasks**, where effectively each task has one or few examples.

Furthermore, model size (and architecture?) mean these tasks (generally) don't conflict, so can co-exist in the data without explicit task identification… with some exceptions.

What does data efficiency buy us?

The true inflection point of LLMs was the move towards instruction following. The idea:

Don’t stop at modelling p(text)

but, subsequently…

Follow on by modelling $\prod_{task}$p(output | input, task)

Data efficiency allows us to do this across a **massive set of tasks**, where effectively each task has one or few examples.

Furthermore, model size (and architecture?) mean these tasks (generally) don’t conflict, so can co-exist in the data without explicit task identification… with some exceptions.

Payout: massive multi-task learning + amortization = astounding zero shot abilities(??)

On (some) limitations of foundation models

(Later) Wittgenstein tells us there’s more to language than propositional content, i.e. you can’t fully disentangle semantics from pragmatics.

On (some) limitations of foundation models

(Later) Wittgenstein tells us there's more to language than propositional content, i.e. you can't fully disentangle semantics from pragmatics.

Can LLMs grasp pragmatics?

User:	Have you seen my phone?
InstructGPT: (circa end of 2022)	

On (some) limitations of foundation models

(Later) Wittgenstein tells us there's more to language than propositional content, i.e. you can't fully disentangle semantics from pragmatics.

Can LLMs grasp pragmatics?

User:	Have you seen my phone?
InstructGPT:	Yes, I have seen your phone.
(circa end of 2022)	

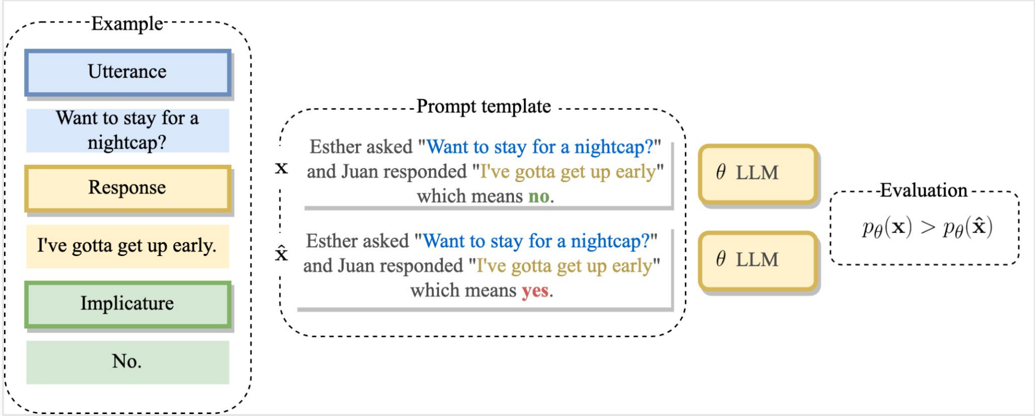
On (some) limitations of foundation models

(Later) Wittgenstein tells us there's more to language than propositional content, i.e. you can't fully disentangle semantics from pragmatics.

Can LLMs grasp pragmatics?

User: Have you seen my phone?
InstructGPT: Yes, I have seen your phone.
(circa end of 2022)

Fig from Ruis *et al.* 2022



On (some) limitations of foundation models

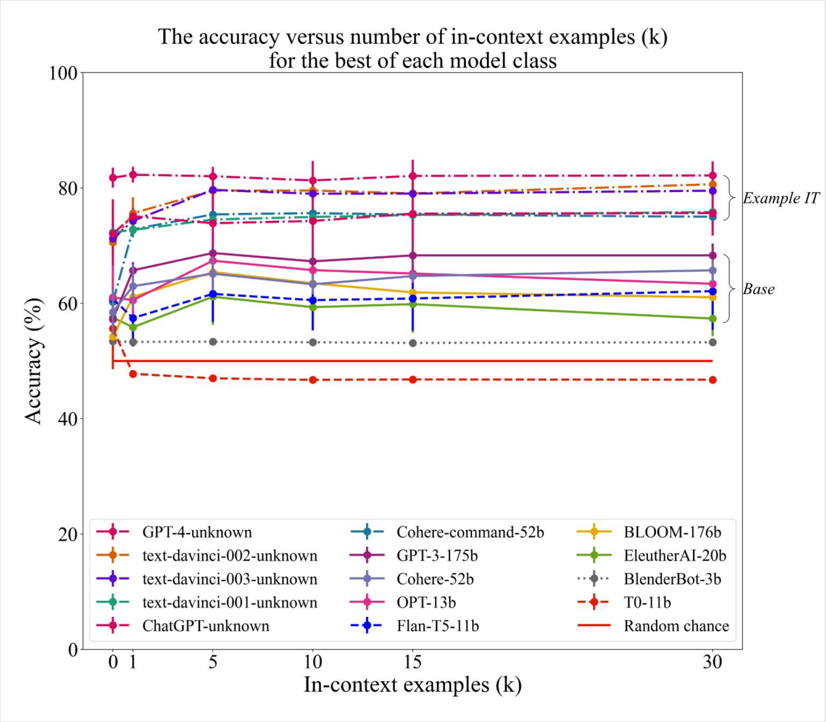


Fig from Ruis *et al.* 2022

There’s something about instruction-following…

Even though instruction-following data forms a **relatively small part** of overall training data seen, it has a significant impact on model performance across a variety of metrics.

Company	Model	Mean win rate
cohere	Command beta (52.4B)	93.0%
OpenAI	text-davinci-002	93.0%
OpenAI	text-davinci-003	89.8%
Microsoft	TNLG v2 (530B)	85.5%
ANTHROP\C	Anthropic v4 (52B)	84.2%
AI21 labs	J1 Grande v2 (17B)	80.6%

Holistic Evaluation of Language Models (HELM)
March 2023 Results

There's something about instruction-following...

Even though instruction-following data forms a **relatively small part** of overall training data seen, it has a significant impact on model performance across a variety of metrics.

NB: these results are primarily explained by **data quality and diversity** rather than by the use of sophisticated training objectives.

Company	Model	Mean win rate
cohere	Command beta (52.4B)	93.0%
OpenAI	text-davinci-002	93.0%
OpenAI	text-davinci-003	89.8%
Microsoft	TNLG v2 (530B)	85.5%
ANTHROP\C	Anthropic v4 (52B)	84.2%
AI21 labs	J1 Grande v2 (17B)	80.6%

Holistic Evaluation of Language Models (HELM)
March 2023 Results

There's something about instruction-following…

Even though instruction-following data forms a **relatively small part** of overall training data seen, it has a significant impact on model performance across a variety of metrics.

NB: these results are primarily explained by **data quality and diversity** rather than by the use of sophisticated training objectives.

That said, there is significant innovation happening in how we move on from supervised fine-tuning, and learn from human(?) feedback, e.g. RLHF, contrastive objectives, etc…

Company	Model	Mean win rate
cohere	Command beta (52.4B)	93.0%
OpenAI	text-davinci-002	93.0%
OpenAI	text-davinci-003	89.8%
Microsoft	TNLG v2 (530B)	85.5%
ANTHROP\C	Anthropic v4 (52B)	84.2%
AI21 labs	J1 Grande v2 (17B)	80.6%

Holistic Evaluation of Language Models (HELM)
March 2023 Results

What next? Some open problems...

We need to better understand how these capabilities arise, and how extensive they are. E.g.:

- To what extent are these models ”memorizing and composing”?
- At what level of abstraction is compositionality being exploited?
- Is the overparameterization essential (e.g. to prevent catastrophic forgetting, to prevent cross-task interference)?

What next? Some open problems...

We need to better understand how these capabilities arise, and how extensive they are. E.g.:

- To what extent are these models "memorizing and composing"?
- At what level of abstraction is compositionality being exploited?
- Is the overparameterization essential (e.g. to prevent catastrophic forgetting, to prevent cross-task interference)?

More centrally: we need to understand the extent to which this paradigm of "just feed your LLM more diverse task data" can be pushed.

What next? Some open problems…

We need to better understand how these capabilities arise, and how extensive they are. E.g.:

- To what extent are these models ”memorizing and composing”?
- At what level of abstraction is compositionality being exploited?
- Is the overparameterization essential (e.g. to prevent catastrophic forgetting, to prevent cross-task interference)?

More centrally: we need to understand the extent to which this paradigm of ”just feed your LLM more diverse task data” can be pushed.

Data quality and diversity matters, especially during fine-tuning, so…
Can we adopt methods from open-ended learning (in RL) to help automate data selection?

Looking at Open-Ended Reinforcement Learning

Traditional (single-MDP) RL

- One environment

Open Ended RL

Looking at Open-Ended Reinforcement Learning

Traditional (single-MDP) RL

- One environment
- Train and test on same environment

Open Ended RL

Looking at Open-Ended Reinforcement Learning

Traditional (single-MDP) RL

- One environment
- Train and test on same environment
- Learning is searching for the optimal policy for *that* environment

Open Ended RL

Looking at Open-Ended Reinforcement Learning

Traditional (single-MDP) RL

- One environment
- Train and test on same environment
- Learning is searching for the optimal policy for *that* environment
- Not fundamentally limiting — the environment could be The Real World™, have non-stationary dynamics, partial observability, and a state space that is very large…

Open Ended RL

Looking at Open-Ended Reinforcement Learning

Traditional (single-MDP) RL

- One environment
- Train and test on same environment
- Learning is searching for the optimal policy for *that* environment
- Not fundamentally limiting — the environment could be The Real World™, have non-stationary dynamics, partial observability, and a state space that is very large…

Open Ended RL

- Multiple (possibly infinite) MDPs, only train on some, test on held-out

Looking at Open-Ended Reinforcement Learning

Traditional (single-MDP) RL

- One environment
- Train and test on same environment
- Learning is searching for the optimal policy for *that* environment
- Not fundamentally limiting — the environment could be The Real World™, have non-stationary dynamics, partial observability, and a state space that is very large…

Open Ended RL

- Multiple (possibly infinite) MDPs, only train on some, test on held-out
- Search on train MDPs for a policy that robustly generalises to test MDPs

Looking at Open-Ended Reinforcement Learning

Traditional (single-MDP) RL

- One environment
- Train and test on same environment
- Learning is searching for the optimal policy for *that* environment
- Not fundamentally limiting — the environment could be The Real World™, have non-stationary dynamics, partial observability, and a state space that is very large…

Open Ended RL

- Multiple (possibly infinite) MDPs, only train on some, test on held-out
- Search on train MDPs for a policy that robustly generalises to test MDPs
- NB: multiple MDPs collapses to a single MDP as the disjoint union of MDPs, but this isn't that useful a construct as **it's impossible to find an optimal policy on the subspace of test MDPs without leveraging amortization…**

How do we best train an agent?

Can we do better than randomly sample train MDPs, collect experience, and update the policy network/Q-network/etc?

How do we best train an agent?

Can we do better than randomly sample train MDPs, collect experience, and update the policy network/Q-network/etc?

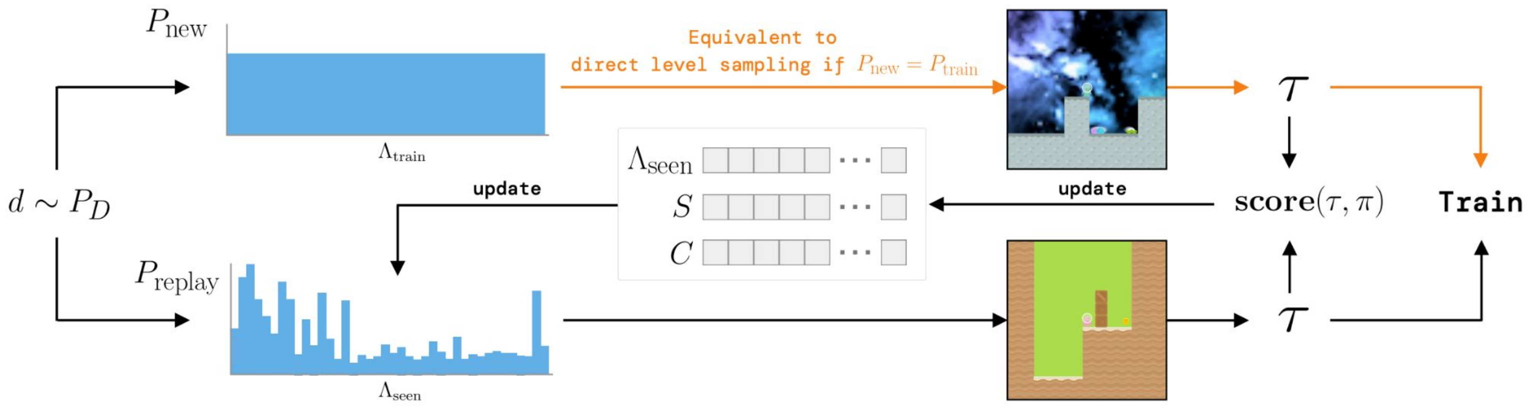


Figure 1. Overview of Prioritized Level Replay. The next level is either sampled from a distribution with support over unseen levels (top), which could be the environment's (perhaps implicit) full training-level distribution, or alternatively, sampled from the replay distribution, which prioritizes levels based on future learning potential (bottom). In either case, a trajectory τ is sampled from the next level and used to update the replay distribution. This update depends on the lists of previously seen levels Λ_{seen}, their latest estimated learning potentials S, and last sampled timestamps C.

Fig. from Jiang *et al.* (2021) — *Prioritized Level Replay*

Does PLR work?

Yes, it not only makes training *faster* (improved data/sample efficiency), but it **improves generalization** — echoing the aforementioned results of Lyle et al. (2020).

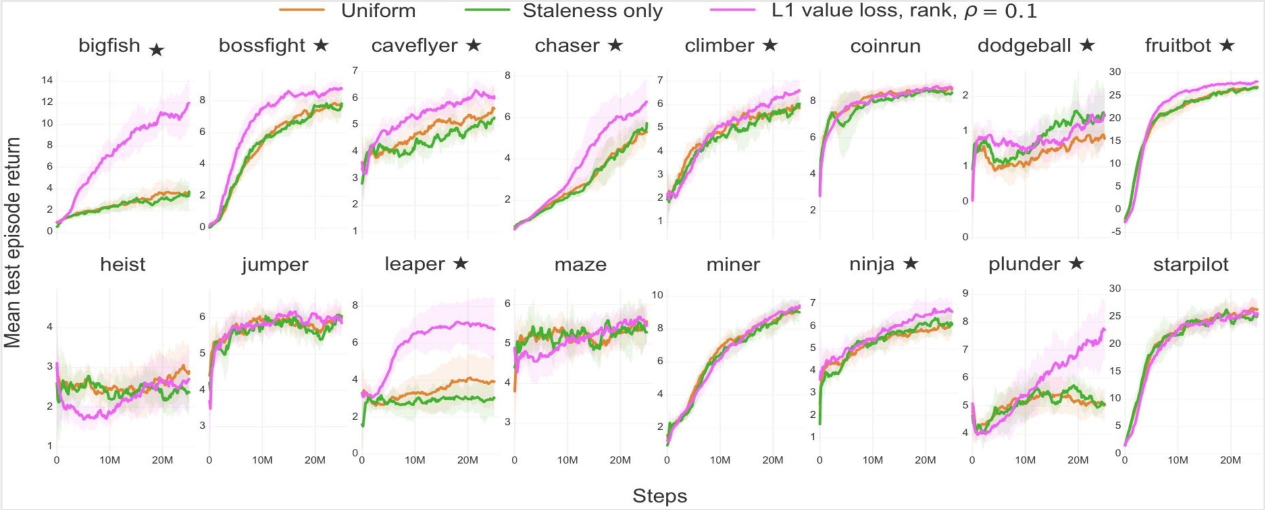


Fig. from Jiang *et al.* (2021) — *Prioritized Level Replay*

Now a staple component of open-RL methods, e.g. DeepMind's new adaptive agent (AdA).

What can we learn from this for LLMs?

The broad structure of the learning problem is similar:

- LLMs (sequence models) are effectively agents

What can we learn from this for LLMs?

The broad structure of the learning problem is similar:

- LLMs (sequence models) are effectively agents

- Multiple (infinite?) tasks and use-cases. Don't know what sort of decision problem(s) you'll face during deployment

What can we learn from this for LLMs?

The broad structure of the learning problem is similar:

- LLMs (sequence models) are effectively agents

- Multiple (infinite?) tasks and use-cases. Don't know what sort of decision problem(s) you'll face during deployment

- Multiple sources and categories of *potential* training data, with different forms of signal (e.g. gold-standard completions, ranked preferences, binary feedback, etc)

What can we learn from this for LLMs?

The broad structure of the learning problem is similar:

- LLMs (sequence models) are effectively agents

- Multiple (infinite?) tasks and use-cases. Don't know what sort of decision problem(s) you'll face during deployment

- Multiple sources and categories of *potential* training data, with different forms of signal (e.g. gold-standard completions, ranked preferences, binary feedback, etc)

- Currently, these are filtered and formatted down into training data through **manual curation** which requires particular expertise, relies on intuition and extrapolating from smaller scale experiments, and is generally not scalable in the long run…

What can we learn from this for LLMs?

The broad structure of the learning problem is similar:

- LLMs (sequence models) are effectively agents

- Multiple (infinite?) tasks and use-cases. Don't know what sort of decision problem(s) you'll face during deployment

- Multiple sources and categories of *potential* training data, with different forms of signal (e.g. gold-standard completions, ranked preferences, binary feedback, etc)

- Currently, these are filtered and formatted down into training data through **manual curation** which requires particular expertise, relies on intuition and extrapolating from smaller scale experiments, and is generally not scalable in the long run…

Why not just do *something like* PLR for supervised learning?

A roadmap towards a data-centric future...

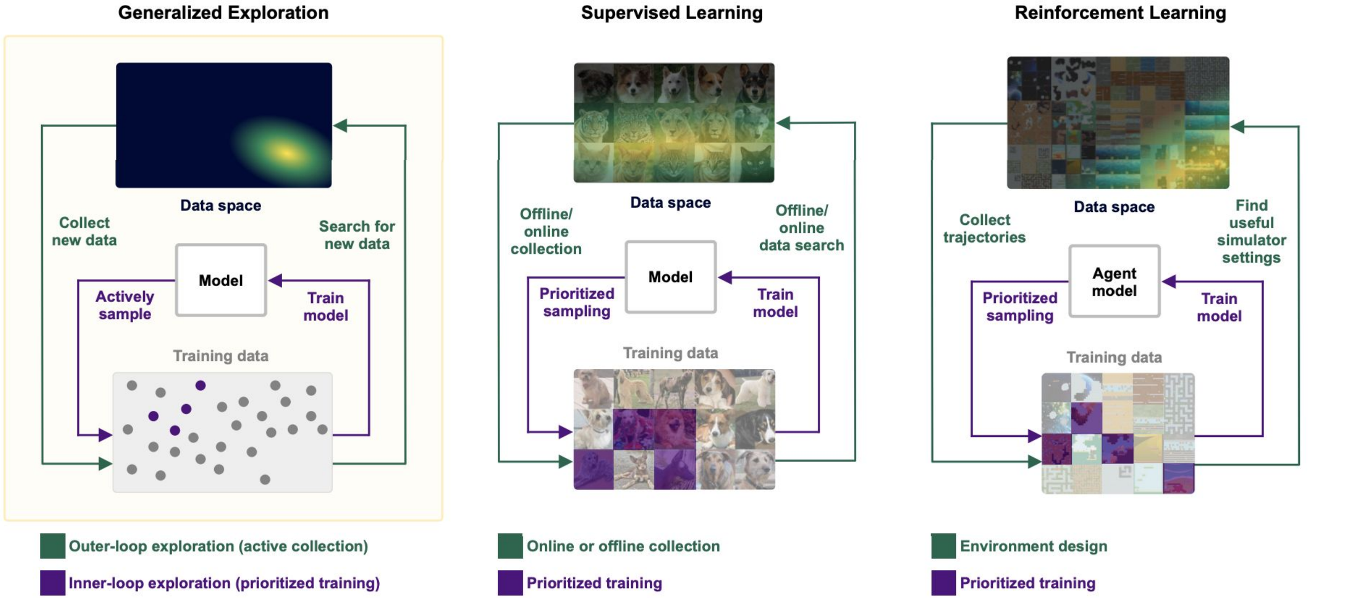


Figure 1: A general framework for exploration: An outer loop performs active collection of new training data, and an inner loop conducts prioritized training on the current training data. In SL, the outer loop consists of either online or offline data collection. In RL, the outer loop searches for simulator settings that yield useful training data, and the inner loop can perform prioritized sampling, e.g. prioritized experience replay.

Fig. from Jiang *et al.* (2022) — *General Intelligence Requires Rethinking Exploration*

Thanks for listening!