Challenges and Applications of Large Language Models

Jean Kaddour[α, †, *], Joshua Harris[β, *], Maximilian Mozes[α],
Herbie Bradley[γ, δ, ϵ], Roberta Raileanu[ζ], and Robert McHardy[η, *]

[α]University College London [β]UK Health Security Agency [γ]EleutherAI
[δ]University of Cambridge [ϵ]Stability AI [ζ]Meta AI Research [η]InstaDeep

Abstract

Large Language Models (LLMs) went from non-existent to ubiquitous in the machine learning discourse within a few years. Due to the fast pace of the field, it is difficult to identify the remaining challenges and already fruitful application areas. In this paper, we aim to establish a systematic set of open problems and application successes so that ML researchers can comprehend the field's current state more quickly and become productive.

Contents

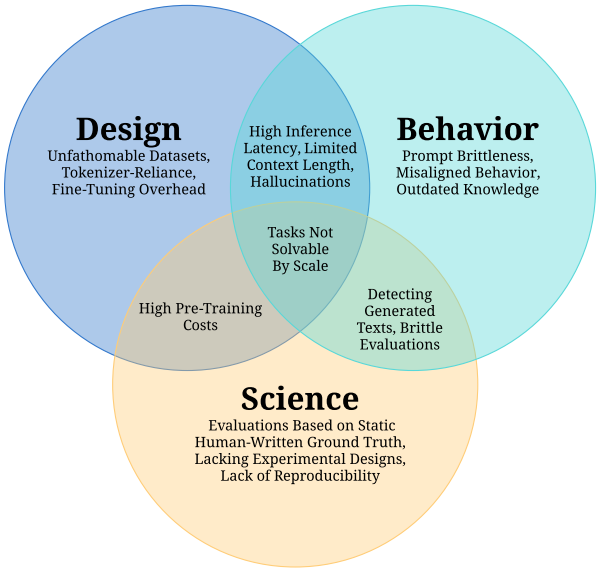


Figure 1: **Overview of LLM Challenges.** *Design*ing LLMs relates to decisions taken before deployment. *Behavior*ial challenges occur during deployment. *Science* challenges hinder academic progress.

1 Introduction

Given the quickly growing plethora of LLM research papers, we aim to address two questions: (1) **Challenges**: What problems remain unresolved? and (2) **Applications**: Where are LLMs currently being applied, and how are the challenges constraining them? For (1), we group the challenges

*Equal contribution.
†{jean.kaddour,robert.mchardy}.20@ucl.ac.uk, joshua.harris@ukhsa.gov.uk

in Fig. 1 into three broader categories "Design", "Behavior", and "Science". To provide answers for (2), we explore the fields of chatbots, computational biology, computer programming, creative work, knowledge work, law, medicine, reasoning, robotics, and the social sciences.

This paper is an opinionated review and assumes familiarity with LLMs and how they work (we refer to more introductory works in Sec. 4). Further, we focus on models trained on text data. We target a technical researcher audience and do not discuss political, philosophical, or moral perspectives on LLMs.

2 Challenges

⚠ Challenge

This box highlights a challenge.

2.1 Unfathomable Datasets

Scaling the amount of pre-training data has been one of the major drivers to equip LLMs with general-purpose capabilities [256]. The size of pre-training datasets quickly outgrew the number of documents most human teams could manually quality-check. Instead, most data collection procedures rely on heuristics regarding data sources and filtering.

In this section, we explore the adverse consequences of these heuristics and the reality that many model practitioners possess only a nebulous understanding of the data on which their model has been trained. We refer to this issue as follows.

⚠ Unfathomable Datasets

The size of modern pre-training datasets renders it impractical for any individual to read or conduct quality assessments on the encompassed documents thoroughly.

Near-Duplicates can arise in different forms and have been reported to degrade model performance [294, 200, 250]. Near-duplicates are harder to find compared to *exact* duplicates; filtering out of such is a standard step in most data collection pipelines, e.g., using the MinHash algorithm [57]. Lee et al. [294] propose the *NearDup* method and find that over 1% of tokens emitted unprompted from a model are part of a memorized sequence of the C4 dataset, e.g., it contains a 61-word sequence repeated $61,036$ times in the training split. By deduplicating it, they reduce the rate of emitted memorizations by 10x. Abbas et al. [6] introduce *SemDeDup*, a technique designed to identify *semantic* duplicates that, although perceptually distinct, convey predominantly similar information, such as sentences with analogous structures with certain words replaced by synonyms. After applying their method to C4, they find that it improves over *NearDup*. Similarly, Kaddour [250] find near-duplicates in the Pile [165] by clustering document embeddings and identifying clusters gathering duplicates.

Benchmark Data Contamination occurs when the training dataset contains data from or similar to the evaluation test set. This can lead to inflated performance metrics, as the model can memorize the test data and simply regurgitate it back during testing.

Finding and removing all training and test data overlaps is difficult in practice. For example, the GPT-3 authors Brown et al. [59] found a code bug after training, resulting in only partially removing all detected overlaps from the training data. They could not afford to retrain the model, so they used it with the remaining overlaps and "cleaned" variants of the considered benchmarks, with all potentially leaked examples removed. They define overlapping examples as examples that share at least 13 consecutive words with any other example in the pre-training set. If an example is shorter than 13 words, they consider it overlapping if it shares all of its words with another example.

Similarly, Dodge et al. [125] search for test data in the web-crawled C4 corpus but measure exact matches, normalized for capitalization and punctuation. They find various input-and-label contaminations of text generation and knowledge completion tasks; and input-only contaminations of the GLUE benchmark. They argue that there are two ways test data can end up in a snapshot of Common Crawl (the original dump source of C4): either a given test set is built from a web text or uploaded after creation. Sainz et al. [472] ask ChatGPT to generate academic benchmark instances, finding that it has memorized multiple ones, including some test splits. Jacovi et al. [237] propose three strategies to mitigate contamination, including encryption and training exclusion controls.

Personally Identifiable Information (PII) such as phone numbers and email addresses, have been found within pre-training corpora, resulting in privacy leaks during prompting. Carlini et al. [65, 67], Lukas et al. [344] extract PII data by prompting GPT-2; Kulkarni [283] report how an engineer yields secret API keys by prompting GitHub Copilot. Henderson et al. [195] discuss the availability of PII in law data across different jurisdictions and filter it based on the legal norm in the respective jurisdiction. El-Mhamdi et al. [137] contend that because strong model performance typically requires memorization of the training data [146, 58], the (undetected) existence of PII in the training data will likely result in models that render them extractable.

Pre-Training Domain Mixtures Several studies have argued for diversity in the pre-training corpus [165, 341, 291]. Many popular corpora follow this by concatenating datasets from different sources, as illustrated in Table 1. However, it remains underexplored what amount of data from different sources is necessary for strong downstream performances. Finding suboptimal mixtures can cause low transferability to downstream tasks [593, 580] and reliance on spurious correlations [253, 618, 347]. Xie et al. [622] find domain mixture proportions by training a small proxy model using group-distributionally robust optimization [471]; surprisingly, they find that the final model trained using their found domain weights yields improved perplexity across all domains, even when it down-weights a domain. Given a target downstream task, Yao et al. [641], Xie et al. [624] select subsets most useful for pre-training. Longpre et al. [341] measure the effects of domain compositions and find that inclusion of heterogeneous data sources is broadly beneficial and likely more important than the data quality (as measured by the document quality classifier employed by PaLM [86] and GLaM [130]) or size, which also motivates smaller yet more diverse pre-training datasets [250].

Fine-Tuning Task Mixtures have to be determined for fine-tuning a pre-trained model on many different tasks, usually with comparatively few examples per task. This technique, which we call multitask-prompted fine-tuned LMs (MTLMs), has demonstrated significant generalization improvements with very little additional training compute.

Date	Name	Size		Sources	Public
		GB	Tokens*		
2014	BookCorpus [684, 36]	5 GB	11 B	Novels	Yes
2019	OSCAR [399]	6.3 T	?	Webpages in 166 languages	Yes
2019	WebText [440]	40 GB	?	Webpages	No
12.2020	CC-100 [100]	2.5 TB	292 B	Webpages in 100 Languages	Yes
12.2020	The Pile [165, 41]	825 GB	300 B	Science, Webpages, GitHub Code, Law, etc.	Yes
2020	C4 [443]	745 GB	156 B	Webpages	Yes
10.2020	mC4 [631]	?	6.3 T	Webpages in 101 Languages	Yes
2021	MassiveText [441]	10.5 TB	2.34 T	Webpages, Books, News, and Code	No
12.2021	GLaM [130]	?	1.6 T	Webpages, Wikipedia, Conversations, Forums, Books, News	No
01.2022	Infiniset [551]	?	2.81 T	Forum dialogs, C4 data, Code, Wikipedia, Webpages	No
06.2022	ROOTS [289]	1.61 TB	2.34 T	Webpages in 46 languages and GitHub Code in 13 languages	Yes
11.2022	The Stack [271]	6 TB	235 B	GitHub Code in 30 languages	Yes
04.2023	LLaMA [556] / RedPajama [98]	2.7 TB	1.2 T	Webpages, GitHub Code, Science, Wikipedia, Books	Yes
06.2023	RefinedWeb [415]	2.8 TB	600 B	Webpages	Yes

Table 1: **Overview of Selected Pre-Training Datasets.** Over the years, pre-training datasets have become more *unfathomable*: they grew rapidly in size and diversity, and not all datasets are publicly available (we do not include datasets that have very little or no information available about them). Unless stated otherwise, the natural language is in English. * We report the number of tokens as provided by the respective paper based on their proposed tokenization scheme.

For example, *instruction fine-tuning* via task instructions prepended to each set of input-output pairs is a very popular scheme, which we will later discuss in more detail in Sec. 2.9. Wang et al. [589] propose `Super-NaturalInstructions`, a fine-tuning dataset with 1,616 diverse tasks and expert-written instructions. Muennighoff et al. [377] extend MTLM to the multilingual setting, showing that fine-tuning on multilingual tasks with English prompts improves results on tasks in all languages.

However, similar to the previous paragraph, how to balance the task datasets well remains unclear.

As the tasks can vary in size considerably, Raffel et al. [443] mix each task in proportion to the number of examples in its 'train' split (up to some `max_num_examples`). Jang et al. [239] report that MTLMs can underperform expert LLMs fine-tuned on only a single task because of (i) negative task transfer, where learning multiple tasks at once hinders the learning of some specific tasks, and (ii) catastrophic forgetting of previous tasks when learning new tasks. Iyer et al. [235] study varying task (sets) proportions, finding several trade-offs and concluding that the right values for these parameters depend on the downstream end-goals. Longpre et al. [340] balance different sets of task sources by omitting them, one at a time, and ranking their contributions on the MMLU benchmark [197]; further, they mix the input prompt templates of zero- and few-shot prompting; finding that this improves the performance in both settings. Another trend is to imitate closed-source models like ChatGPT by collecting a dataset of API outputs (against OpenAI's terms and conditions) and fine-tuning an open-source LM with it [540]. However, Gudibande et al. [180] point out that such imitation models are only good at mimicking the proprietary model's style but not its content, a distinction that has been discussed extensively in the causality literature [253]. They conclude that substantial capability gaps between fine-tuned open-sourced and closed-source models remain, motivating future work for better imitation data.

2.2 Tokenizer-Reliance

Tokenization is the process of breaking a sequence of words or characters into smaller units called tokens, such that they can be fed into the model. One common tokenization approach is *subword tokenization*, where we split words into smaller units, called *subwords* or *WordPieces* [490]. The goal is to handle rare and out-of-vocabulary words in a model's vocabulary effectively while maintaining a limited number of tokens per sequence in the interest of computational complexity. Subword tokenizers are usually trained unsupervised to build a vocabulary and optionally merge rules to encode the training data efficiently.

However, the necessity of tokenization comes with multiple drawbacks [257]; some of which we discuss below. For example, Ahia et al. [13], Petrov et al. [426] show that **the number of tokens necessary to convey the same information varies significantly across languages**, making the pricing policy of API language models, which charge users based on the number of processed or generated tokens, potentially unfair. They find that users of many supported languages are overcharged while receiving subpar results, with this group predominantly residing in areas where these APIs are already less affordable.

Further, **discrepancies between the data that a tokenizer and a model have been trained on can lead to glitch tokens** [465], which can subsequently cause unexpected model behavior as their corresponding embeddings are essentially untrained. This coupling between the tokenizer and pre-training corpus creates the burden of a new training run of the tokenizer each time the pre-training corpus is modified.

Next, Tokenization schemes that work well in a multilingual setting, particularly with non-space-separated languages such as Chinese or Japanese, remain challenging [157, 91].

Existing subword tokenization schemes are predominantly greedy algorithms trying to encode language as efficiently as possible regarding the number of tokens used. Naturally, these methods favor subwords comprising larger parts of the training data and, therefore, subwords that are shared across many languages. This favors languages with shared scripts like Latin and Cyrillic, resulting in suboptimal tokenization of low-resource languages [92, 676].

⚠ Tokenizer-Reliance

Tokenizers introduce several challenges, e.g., computational overhead, language dependence, handling of novel words, fixed vocabulary size, information loss, and low human interpretability.

Subword-Level Inputs are the dominant paradigm, providing a good trade-off between vocabulary size and sequence length. **Byte-Pair Encoding [490, 577]** (BPE) starts with the set of symbols (characters or bytes) that comprise the training data. The tokenizer is then trained to learn rules to merge the most frequent pair of two consecutive tokens—defined by the existing vocabulary—into a new vocabulary item. Byte-level BPE (BBPE) [577] is an extension of BPE with byte-level subwords, particularly

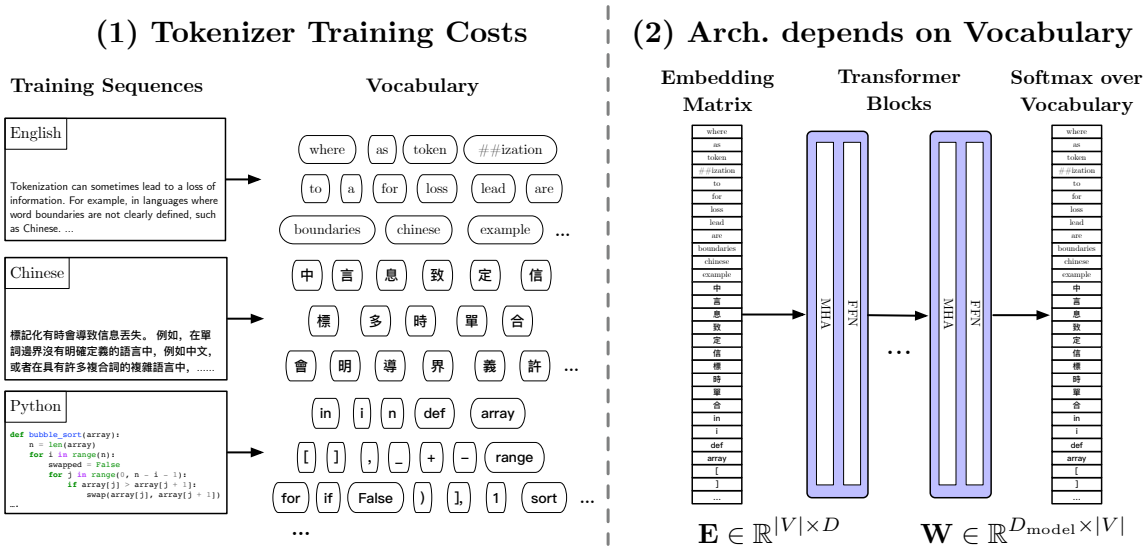


Figure 2: **Exemplary Drawbacks of relying on Tokenization.** (1) The tokenizer training step involves non-trivial computations, e.g., multiple passes over the entire pre-training dataset, and introduces a dependency on it, which can become especially problematic in multilingual settings. (2) The embedding layer $\mathbf{E}$ and output layer $\mathbf{W}$ of LLMs involve the vocabulary size; e.g., making up $\approx 66\%$ of the model's parameter count in T5 models [629].

suited for multilingual tasks where it enables vocabulary sharing between languages. A trained BPE tokenizer applies the previously learned rules to tokenize inputs. **WordPiece [485, 617]** is a closed-source tokenization algorithm used, e.g., in BERT [120]. Like BPE, WordPiece starts with a small initial vocabulary, which is iteratively extended by learning merge rules and creating new vocabulary items. Rather than selecting the most frequent pair of consecutive tokens, WordPiece uses a scoring function to normalize the frequency of the pair by the frequencies of the individual tokens to prioritize common pairs with rare individual tokens. **Unigram Tokenization [281]** iteratively trims a large base vocabulary to a given target size. To this end, at each step of the tokenizer training, a unigram language model is used to compute a loss over the training data conditional on a certain vocabulary item being removed. A proportion of the subwords with the lowest losses are removed to form the base vocabulary for the next iteration. Unigram tokenization is probabilistic, i.e., during inference, all possible tokenizations of a given sequence are scored using the unigram language model, and the most likely one is selected. **SentencePiece [282]** is a commonly used open-source library, implementing several tokenization algorithms such as (B)BPE and Unigram tokenization. SentencePiece also implements non-subword tokenization approaches like word- and character-level tokenization.

Byte-Level Inputs are an alternative to subword tokenization is use byte-level inputs. Byte-level inputs can either be used in combination with subword tokenizers [577] or used to define a limited vocabulary that can be used to encode all possible sequences. For example, Xue et al. [630] train a non-subword mT5 model using UTF-8 bytes rather than subword tokens as inputs, showing promising performance on multilingual data. While this enables subword-free LLMs, UTF-8 encodes Latin languages with fewer bytes than e.g., Chinese, Japanese or Korean[1]. Tay et al. [546] propose the Charformer, a tokenization-free model which learns a soft subword tokenization in latent space (Gradient-Based Subword Tokenization) given byte-level inputs. Charformer performs comparably to subword-based models while incurring less computational overhead than other byte or subword models. Choe et al. [83] train a small-scale, 0.8B language model based on raw byte-level inputs and show that it performs comparably. On a smaller scale, Clark et al. [94] show that their tokenization- and vocabulary-free encoder *Canine* outperforms a comparable tokenization-based model. Yu et al. [652] address the computational cost that byte-level tokenization incurs by segmenting input sequences into local patches, which can be processed in parallel. Similarly, Horton et al. [212] propose to operate directly on file bytes. In a

[1] https://www.unicode.org/versions/Unicode15.0.0/

parallel line of work, Rust et al. [467] render text as images and train an encoder model to predict the raw pixels of the images.

2.3 High Pre-Training Costs

The vast majority of the training costs go toward the pre-training process. Training a single LLM can require hundreds of thousands of compute hours, which in turn cost millions of dollars and consume energy amounts equivalent to that used by several typical US families annually [412, 86, 44]. Recently proposed scaling laws [256] posit that model performances scale as a power law with model size, dataset size, and the amount of compute used for training, which is fairly unsustainable and can be classified as Red AI [487], where state-of-the-art results are essentially "bought" by spending massive computational resources. For example, depending on the exact law coefficients, reducing the error from 3% to 2% can require an order of magnitude more data or compute [518].

⚠ Unsustainable Loss Power-Law [256]

Performance increases through larger compute budgets but at a decreasing rate if the model or dataset size is fixed, reflecting a power law with diminishing returns.

In the following, we look at two lines of work aiming at resolving such issues.

Compute-Optimal Training Recipes [201, 256] In Sec. 2.1, we discussed how the availability of LLM pre-training data has become abundant through the quickly-spread practice of including web-crawled text. Further, thanks to the introduction of Transformer models [563] and suitable hardware [210], we have scaled models to unprecedented sizes. Assuming that we have not yet reached the limits of data [45, 568, 415] nor model sizes [256, 206, 398]; currently, the main bottleneck is the amount of compute available [1]. Given a particular budget, how large should the pre-training corpus and model be to maximize training efficiency?

As mentioned at the beginning of this section, one recent proposal is to learn empirical "*scaling laws*" [201, 256], which describe the relationship between LLM performance and the compute budget, model, and dataset size. These laws can provide the right scaling recipe for compute-optimal training, ideally, even when extrapolating to larger compute budgets. For example, OpenAI [398] report that they were able to accurately predict the model performance of the full-size GPT-4 model based on the performance of a series of smaller models using at most 10,000x less compute than the full model.

The exact power law coefficients are still heavily debated. Kaplan et al. [256] put forward that the model size should be scaled more aggressively than the dataset size to use a given compute budget optimally. Contrary to this, Hoffmann et al. [206] find that many LLMs are undertrained and argue that the number of parameters and data should be scaled equally. However, power laws sometimes come in the form of bounds, which can span an order of magnitude difference in the amount of data to be used given a concrete compute budget [665]. Further, the pre-training loss does not always correlate well with downstream performance [252, 332, 251].

The viewpoint of Touvron et al. [556], Vries [571], Touvron et al. [557] is that when selecting a model size, the computation resources for later usage (inference) should be considered, not just the one-time training costs. They suggest that it might be beneficial to train a smaller model more intensively upfront to offset larger inference costs in the future. Hence, they train models of various sizes on more tokens than are typically used to achieve the best performance possible, given the model size.

One remaining hurdle of performance prediction is inverse scaling, which we discuss in Sec. 2.14. Since scaling laws were typically constructed in the context of pre-training and thereby decoupled from downstream tasks, it remains an open question of how to predict inverse scaling properties. Tay et al. [544] find that scaling laws can differ in upstream and downstream setups; aside from only the model size, model shape matters for downstream fine-tuning.

Pre-Training Objectives Various pre-training objectives (PTO) are suitable for performing self-supervised training of LLMs. The exact choice of PTO heavily influences the model's data efficiency during pre-training, which in turn can reduce the number of iterations required. A PTO typically is a function of the (i) architecture, (ii) input/targets construction (e.g., target span length, low/high corruption, see Fig. 4), and (iii) masking strategy (Fig. 3). While (i) and (ii) can be disentangled and

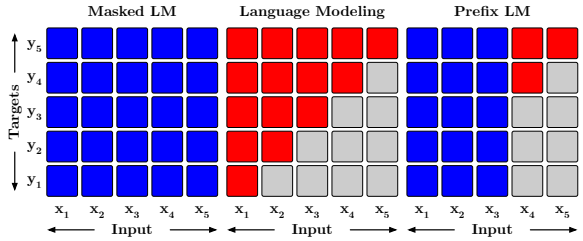


Figure 3: **Masking Strategies.** Each row denotes to which inputs $\boldsymbol{x}_i$ (columns) a particular output $\boldsymbol{y}_i$ (row) can attend to (uni- or bi-directional).

should not be conflated conceptually [545], in practice, there exist popular combinations that achieve good performances.

Attending to all tokens, as shown in Fig. 3(left), is the most data-efficient strategy since it uses context from before and after the token to be predicted. However, for that reason, it is unsuitable for text generation [120], since it considers future context for prediction. We typically employ it in natural language understanding (NLU) tasks [120], where it has shown strong results. The next token prediction objective is most suitable for natural language generation (NLG) but also the least data efficient since it only attends to the past context (Fig. 3(middle)). More recent advances in pre-training objectives aim to find a middle-ground to increase data efficiency by providing stronger and more diverse training signals, e.g., the Prefix LM, which partly attends to past tokens, as illustrated in Fig. 3(right) and discussed below.

The following discusses the trade-offs between some of the recently proposed objectives. Fig. 4 visually depicts the different pre-training objectives. Notation-wise, we denote a sequence of N tokens x as $x = x_1, \ldots, x_N$.

We start with the most basic and still widely-used **Language Modeling [59]** (or *next token prediction*) objective. Here, we learn parameters $\boldsymbol{\theta}$ by maximizing the likelihood of the next token given the previous tokens,

$$L(x) = \sum_{i=1}^{N} \log P(x_i | x_1, \ldots, x_{i-1}; \boldsymbol{\theta}). \quad (1)$$

Masked Language Modeling (MLM; or Cloze) [549, 120] hides a set proportion of tokens in the sequence by replacing them with a special [MASK] token. The literature employs the MLM objective for non-autoregressive, i.e., non-generative, bidirectional context models, where the model uses tokens before and after the target token for predictions, leveraging a more holistic understanding of its context than the NTP objective. Furthermore, we can use each input sentence to predict multiple masked tokens in a single pass, while the NTP objective typically learns from predicting one token at a time.

Let x_{MASK} denote the set of indices of the masked tokens and $x_{\neg\text{MASK}}$ the unmasked tokens. The objective of MLM is then to maximize the likelihood given the parameters $\boldsymbol{\theta}$,

$$L(x_{\text{MASK}} | x_{\neg\text{MASK}}) = \frac{1}{|x_{\text{MASK}}|} \cdot \sum_{i \in x_{\text{MASK}}} \log P(x_{\text{MASK}_i} | x_{\neg\text{MASK}}; \boldsymbol{\theta}). \quad (2)$$

Patel et al. [410] show that such models produce representations more suitable for transfer learning; however, they come with difficulties in performing in-context learning (Sec. 2.7).

To further improve the training efficiency of the MLM objective, Bajaj et al. [33] propose to replace input tokens with ones generated by an auxiliary language model (ALM), resulting in a *Model generated dEnoising TRaining Objective* (METRO). Their approach consists of roughly three components: (i) train an ALM using the MLM objective, (ii) given some inputs with masked positions, predict the tokens (with the ALM), (iii) train the main model to correct these tokens inserted in the masked positions, i.e., 1) predict whether the ALM has replaced a token and if so, 2) predict the original token. They train the auxiliary and main model jointly.

Prefix Language Modeling [443] generalizes language modeling by allowing prefix tokens with a bidirectional receptive field to be added to the input (without prefix, it is equivalent to standard LM). Note that this is still different from the bidirectional context as in MLM, where we always condition on all the tokens before and after the masked ones (see Fig. 3 left). For computing the hidden states of the prefix, prefix-LM attends to tokens before and after (see Fig. 3 right).

Span Corruption [303, 443, 132] or *span denoising* refers to a group of denoising objectives that generalize MLM to denoise contiguous sequences of tokens within a given text, called *spans*. The denoising objectives typically replace the sampled spans with a single unique masking token and train the model to fill it in. Raffel et al. [443]

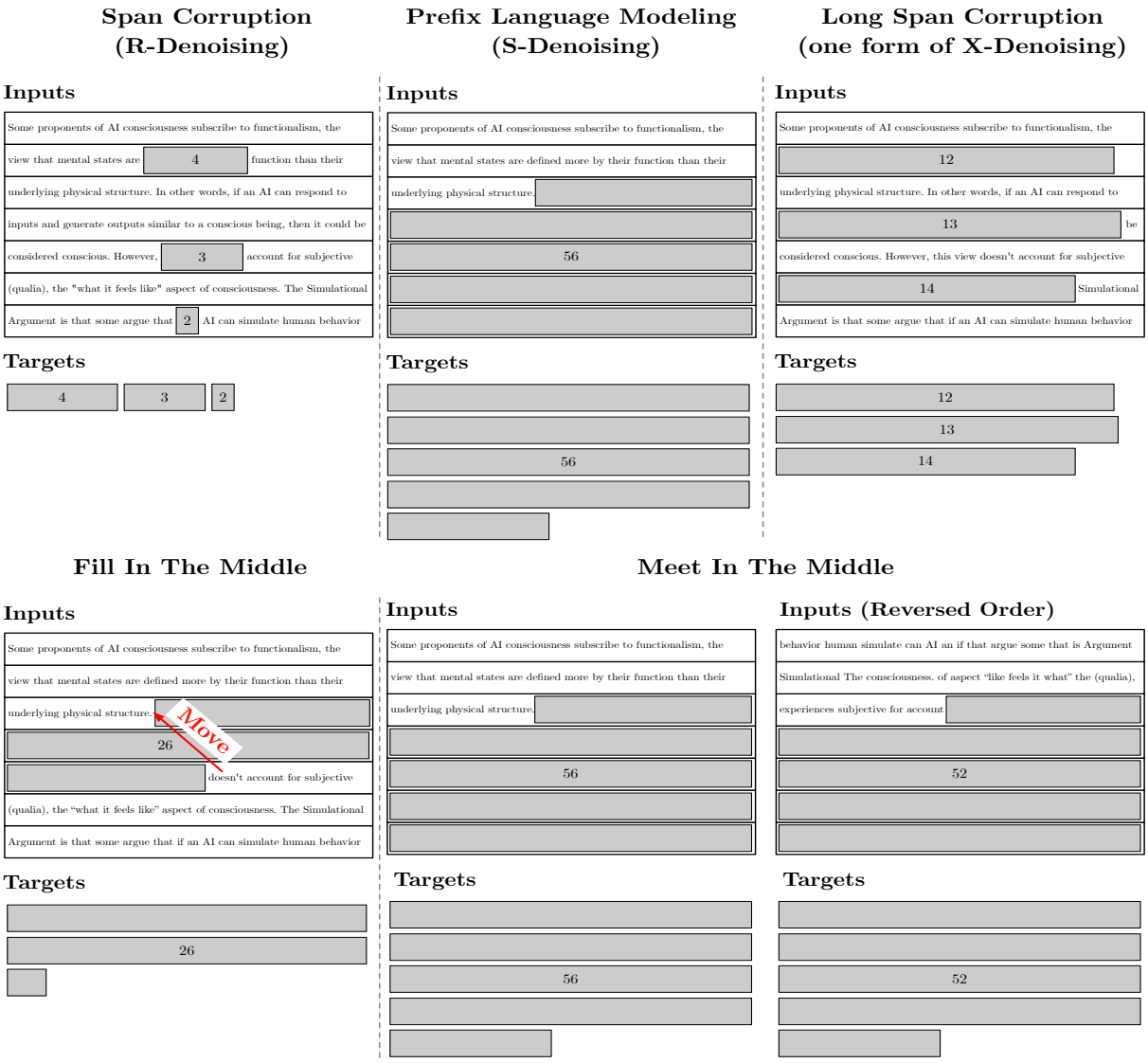


Figure 4: **Self-Supervised Data Construction by Pre-Training Objectives**, adopted from Tay et al. [545]. We indicate masked tokens with gray rectangles, which become the targets. For brevity, we omit special tokens.

shows that this can speed up training because span corruption produces shorter sequences on average compared to corrupting individual tokens in an i.i.d. manner.

Mixture of Denoisers [545] (MoD) refers to injecting objective diversity by mixing multiple denoising objectives. Tay et al. [545] categorize three denoising objectives: {R,S,X}-Denoiser. The **r**egular denoising corresponds to the previously introduced span denoising. **S**pecific denoising comprises splitting a given sequence into a prefix acting as the context and a suffix acting as the target. In e**x**treme denoising, we corrupt large parts of the input by either (a) increasing the proportion of masked tokens per span or (b) increasing the span length forcing the model to generate long sequences with limited context, which we illustrate in Fig. 4). The MoD objective has subsequently been shown to improve model performance by continuing training pre-trained LLMs [443, 86] for relatively few steps [547].

Fill In the Middle Bavarian et al. [38] propose to augment the next token prediction objective by shuffling tokens within a document such that we *fill in the middle* (FIM) based on prefix and suffix. They demonstrate that models pre-trained on a mixture of FIM-transformed and left-to-right data result in left-to-right and FIM capability models.

Meet in the Middle Nguyen et al. [382] extend the FIM objective by enabling bidirectional context to construct a denser, more data-efficient supervision signal while maintaining the autoregressive

nature of the underlying model: They train two decoders—one forward $\overrightarrow{p}\left(x_i \mid x_{<i}; \boldsymbol{\theta}\right)$ and one backward language model $\overleftarrow{p}\left(x_i \mid x_{<i}; \boldsymbol{\theta}\right)$—with shared parameters $\boldsymbol{\theta}$. Additionally, they add an agreement regularize to the loss, encouraging the forward and backward model to agree: for a dataset S of sequences, the full pre-training loss is

$$\sum_{x \in S} \sum_{i=1}^{|x|} \underbrace{-\log \overrightarrow{p}\left(x_i \mid x_{<i}; \boldsymbol{\theta}\right)}_{\text{NLL for forward model}} \underbrace{-\log \overleftarrow{p}\left(x_i \mid x_{>i}; \boldsymbol{\theta}\right)}_{\text{NLL for backward model}} \underbrace{+\beta D_{i,x}^{TV}\left(\overrightarrow{p} \| \overleftarrow{p}\right)}_{\text{agreement regularizer}}, \tag{3}$$

where $D_{i,x}^{TV}\left(\overrightarrow{p} \| \overleftarrow{p}\right)$ is the total variation distance among the two models on the i-th token. Once pre-training has been completed, we can use only the forward model $\overrightarrow{p}$.

Parallelism Strategies The sheer size of LLMs makes it hard to train or even do inference with them on only one accelerator (GPU, TPU, etc.). A common solution is *model parallelism*, which can be viewed as a *divide-and-conquer* strategy: we slice up various parts of the model (dividing the problem into sub-problems), distribute them across multiple devices, with each device computing a portion of the overall computation (solve each problem independently) and combine all results to produce the final output (forward/backward pass).

Implementing model parallelism synchronously creates a problem where running data batches through multiple workers with sequential dependency (each layer depends on results from the previous layer) leads to significant waiting times and under-utilization of computation resources.

Another strategy is *pipeline parallelism*, which combines model parallelism with *data parallelism*, meaning that we not only distribute parts of the model across different devices but parts of the data too, i.e., each worker splits its mini-batch further into micro-batches with gradients being accumulated across all micro-batches before the weight update. Huang et al. [226] instantiate such an approach called *GPipe*, which divides each mini-batch into smaller micro-batches distributed across different accelerators simultaneously; gradients are applied synchronously at the end. Compared to naive model parallelism, this decreases waiting times and increases the utilization of computational resources.

These issues have motivated asynchronous parallelization schemes. Recht et al. [453] present *Hogwild!*, which *greedily* applies gradients to the local weights on each accelerator as soon as they arrive, offering better resource utilization than pipeline parallelism but suffering from training instabilities due to *stale gradients* which are based on outdated model weights.

Gomez et al. [172] propose *N-Wise interlocking backpropagation*, which is a generalization of end-to-end and local training. While end-to-end (global) training performs a forward pass through all layers, computes a loss and gradients, and backpropagates through all layers, local training performs forward passes through all layers individually and immediately computes a local loss and gradient update, offering higher resource utilization at the cost of (empirically) worse task performance. *N-Wise interlocking backpropagation* strikes a compromise by performing a forward pass through N layers before computing a loss and updating the parameters of the associated layers, enabling better layer communication than local training and higher computational efficiency than end-to-end training.

Chowdhery et al. [86] leverage a combination of model parallelism and fully sharded data parallelism (FSDP) [628, 674]—a technique where each device only holds a subset of the model parameters, gradients, and optimizer states, and parameters necessary for local computations are communicated on-demand—to enable highly parallel, high throughput training across thousands of chips within a single TPU pod. PaLM further employs data parallelism to achieve scaling at pod level, leveraging the Pathways [37] system to distribute data.

In a parallel line of work, Lepikhin et al. [298] propose *GShard*, a model parallelism method that extends the XLA [468] compiler, enabling automatic sharding of models.

Miscellaneous Rae et al. [441] stack the layers of a 4.5B parameter model to jump-start and accelerate the training of a 9B model, which led to a 40% reduction in compute; an idea that has been previously used for training smaller-scale LMs [173]. Brown et al. [59] progressively increase the batch size from a small to the full value over training when training GPT-3; a trick that has been previously used for training image mod-

els [514]. Sanyal et al. [476] apply latest weight averaging [249] to LLMs between 1 and 12B parameters; for a 6.9B parameter model, they reach savings of up to 4,200 GPU hours. For smaller-scale models, there exist various pre-training speedup algorithms [663, 685], but they have not been scaled up yet and shown to offer only limited gains when compared with budget-adjusted baselines [251].

2.4 Fine-Tuning Overhead

A potential drawback of pre-training LLMs on massive and diverse sets of textual data is that the resulting models might struggle to explicitly capture the distributional properties of task-specific datasets. To address this, fine-tuning refers to adapting the pre-trained model parameters on comparatively smaller datasets that are specific to an individual domain or task. LLM fine-tuning is highly effective at adapting LLMs for downstream tasks [215, 120, 440].

Technically speaking, fine-tuning can be achieved by further training a model on a smaller dataset. Depending on the model architecture, this is done by either (i) directly fine-tuning pre-trained models using a standard language modeling objective or (ii) adding individual learnable layers to the output representations of a pre-trained language model, which are designed to create compatibility between the model's output representations and the output formats of individual downstream tasks (e.g., for text classification or sequence labeling). See Devlin et al. [120] (Figure 1) for an illustration.

However, LLMs with billions of parameters have large memory requirements to store (i) the model parameters, (ii) the model activations, and (iii) the gradients and corresponding statistics. Due to limited device memory (e.g., GPU or TPU) necessitates access to large clusters with many devices to fine-tune a full LLM, limiting access to a few institutions with large compute resources.

⚠ Large Memory Requirements

Fine-tuning entire LLMs requires the same amount of memory as pre-training, rendering it infeasible for many practitioners.

Moreover, while full model fine-tuning is effective at adapting LLMs to perform well on specific downstream tasks, individual copies of fine-tuned LLMs need to be stored and loaded for individual tasks, which is computationally inefficient [213, 311] and requires practitioners to keep individual fine-tuned LLMs in memory for every task. We illustrate this overhead in Figure 5.

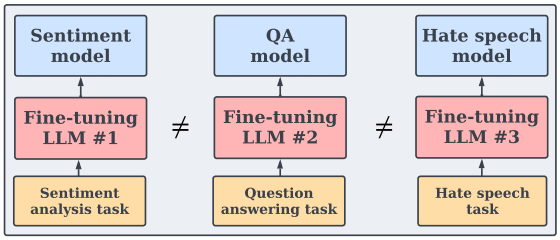
⚠ Overhead of Storing and Loading Fine-Tuned LLMs [213, 311]

When adapting an LLM via full-model fine-tuning, an individual copy of the model must be stored (consuming data storage) and loaded (expending memory allocation, etc.) for each task.

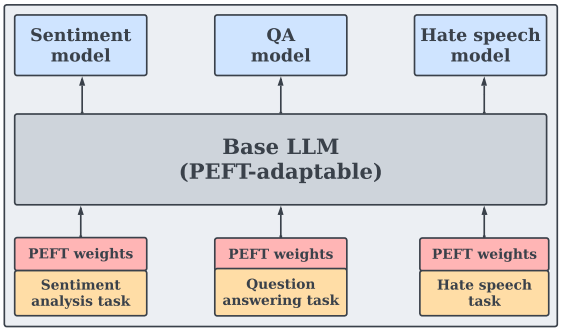
Parameter-efficient fine-tuning An alternative method to adapt an LLM to a specific dataset/domain is via *parameter-efficient fine-tuning* (PEFT). PEFT refers to a class of methods that adapt LLMs by updating only a small subset of model parameters. **Adapters [213]** are one of the earliest works on PEFT. This method incorporates additional, learnable layers into a Transformer architecture that are updated during fine-tuning whilst keeping the remainder of the network unchanged. Experimental results on 26 text classification tasks (incl. the GLUE benchmark [575]) reveal that models trained via Adapters are competitive with full fine-tuning while updating only 3% of the model's parameters. Ben Zaken et al. [40] instead propose only to update the model's bias terms for fine-tuning, which make up less than 1% of the model's parameters. Experimental results show competitive performance across tasks of the GLUE benchmark. We are aware of three general frameworks for incorporating adapters into language model fine-tuning, namely AdapterHub [428], LLM-Adapters [219], and HuggingFace's PEFT library [356].

PEFT methods introduced for larger models include **prefix-tuning** [311] and **prompt-tuning** [299], which both operate by prepending a set of learnable token embeddings to an input. These token embeddings (also referred to as *soft prompts* [299]) are learned during the fine-tuning stage, whereas the remainder of the model parameters remains fixed. Most notably, such soft prompts contain thousands rather than millions of parameters and are much more efficient to store. Notably, one still has to backpropagate through the network while fine-tuning the tokens. Alternatives for models with only black-box API access have been proposed too [528, 122].

It has been shown that prompt-tuning can learn generalizable representations with very small



(a)



(b)

Figure 5: **Fine-tuning an LLM for a specific downstream task**. (a) illustrates vanilla fine-tuning, which requires updating the entire model, resulting in a new model for each task. In (b), PEFT instead learns a small subset of model parameters for each task with a fixed base LLM. The same base model can be re-used during inference for different tasks.

amounts of training data, achieving competitive performances when trained on less than 100 examples for safety classification [376] or five examples for multilingual question answering [11]. In addition to that, recent work investigates the potential of using soft prompts for pre-training and transfer learning across different tasks [179, 572].

Liu et al. [331] introduce $(IA)^3$, which scales activations in individual Transformer layers with learnable vectors. The authors demonstrate its effectiveness by showing that models trained using $(IA)^3$ outperform full model fine-tuning on various datasets whilst updating only 0.01% of the model's parameters.

Malladi et al. [355] propose a memory-efficient zeroth-order (MeZO) optimizer, which only requires the same memory footprint as during inference (instead of storing gradients or optimizer states). Further, it can optimize non-differentiable objectives like accuracy or F1 scores, which conventional gradient-based tuning methods cannot.

Hu et al. [218] propose Low-Rank Adaptation (LoRA), which formulates parameter updates of weight matrices at individual Transformer layers as an additive low-rank decomposition. Such a reparameterization avoids the need to compute dense matrix multiplications. Dettmers et al. [118] extend LoRA to quantized LLMs, drastically reducing memory usage, allowing them to fine-tune a 65B model on a single 48GB GPU. The authors mention that regular training of the same model requires more than 780 GB of GPU memory.

Compute Requirements However, despite substantial improvements in *memory complexity* needed to fine-tune LLMs for specific tasks, a remaining challenge is the *time complexity*. Fine-tuning an LLM, even with PEFT methods, still requires full gradient computation. The computational infrastructure needed to adapt LLMs prohibits potential applications like personalization on smaller devices.

⚠ Full Matrix Multiplications

Parameter-efficient fine-tuning of LLMs still requires computing full forward/backward passes throughout the whole network.

2.5 High Inference Latency

According to Pope et al. [431], Weng [605], two reasons why LLMs exhibit high inference latencies are: (1) **low parallelizability** since the inference procedure proceeds one token at a time and (2) **large memory footprints**, due to the model size and the transient states needed during decoding (e.g., attention key and value tensors). Further, the authors also discuss the quadratic scaling of the attention mechanisms in Transformers, which we discuss separately in Sec. 2.6.

⚠ High Inference Latency [431, 605]

LLM inference latencies remain high because of low parallelizability and large memory footprints.

In the following section, we review techniques used to address these challenges by e.g., reducing the memory footprint (size and/or bandwidth), or accelerating specific computational operations. Note that some of these techniques may also be applicable during the training process, but we discuss them here since they are not only designed for training, like the approaches discussed in Sec. 2.3.

Efficient Attention Roughly two lines of work aim to accelerate attention mechanism computations by (i) lower-level hardware-aware modifications or (ii) higher-level sub-quadratic approximations of the attention mechanism.

For the former, multi-query attention [493] aims to reduce memory bandwidth bottlenecks when sequentially generating sequences of tokens using Transformer decoder layers by keeping only one attention head for the key and value tensors. Similarly, Dao et al. [107], Pagliardini et al. [404] reduce memory bandwidth by proposing an alternative computation method for multi-head self-attention, called `FlashAttention`, to minimize the number of I/O operations to speed up the computation on modern GPUs. As an optimized attention implementation, `FlashAttention` leverages operator fusion to reduce the memory bandwidth bottleneck. Pagliardini et al. [404] build on top of `FlashAttention` and incorporate attention sparsity patterns, encompassing key/query dropping and hashing-based attention. Pope et al. [432] implement different sharding techniques to efficiently spread the feedforward and attention computations across devices while optimizing for inter-device communication costs, enabling context lengths of up to 43,000 tokens using multi-query attention.

With regards to the second stream of work, a common theme to improve the computational or memory complexity of the attention mechanism is to sparsify the attention matrix or introducing (linear) approximations [543]. However, the scalability of some efficient Attention approximations has been questioned. For example, Tay et al. [542], Hua et al. [220] find that the Performer attention approximation [85] severely underperforms the vanilla self-attention mechanism, especially when scaled up to large models.

Quantization is a post-training technique that reduces the memory footprint and/or increases the model's throughput by reducing the computational precision of weights and activations. `nuQmm` [407] and `ZeroQuant` [643] use a non-uniform quantization method to quantize weights and apply custom CUDA kernels for computational benefits. `LLM.int8()` [117] is a degradation-free quantization scheme enabling efficient inference of multi-billion parameter LLMs by utilizing Int8 quantization and falling back to higher precision for certain outlier features without the need for re-training. Similarly, GLM-130B [658] uses a degradation-free 8-bit quantization scheme, storing weights in 8-bit and performing matrix multiplications in 16-bit precision. Frantar et al. [153] propose an efficient, one-shot quantization technique to compress LLM weights down to 3 to 4 bits per weight, enabling 175B parameter models to be run on a single GPU. Dettmers et al. [119] further improve upon this by combining higher precision representations for outlier weights and grouped quantization.

Pruning is a complementary post-training technique to quantization, removing parts of the weights of a given model (without degrading its performance). An important distinction is whether the pruning follows a *structured* pattern or is *unstructured*. Structured sparse models substitute dense sections of a model with an assembly of significantly smaller yet still dense components. Unstructured sparse models contain weights of value zero, which do not influence the network's behavior and can therefore be committed in theory. However, in practice, it is more challenging to translate theoretical to practical computation savings on current hardware [161, 112, 336].

On the structured side, early work on pruning language models mainly aims at comparatively small MLM-type models [592, 143, 243]. Ma et al. [349] propose LLM-Pruner, which aims at pruning LLMs in a task-agnostic manner while preserving the zero-shot capabilities of the models. To this end, LLM-Pruner adopts a three-stage pruning procedure where 1) interdependent structures within the model are identified and grouped, 2) the contribution to the overall performance is estimated for each group, and low-performing groups are pruned, 3) performance recovery via parameter-efficient fine-tuning procedure using LoRA [218].

On the unstructured side, SparseGPT [152] is an unstructured pruning approach specifically developed to be fast enough to be run on LLMs with hundreds of billions of parameters within a few hours, being able to prune the number of parameters by up to 60% while maintaining roughly the same model performance. Sun et al. [527] propose Wanda (Pruning by **W**eights **and a**ctivations), which applies magnitude pruning based on the product of each weight's magnitude and the norm of the corresponding input activations, matching SparseGPT in performance while requiring only a single forward pass to prune the network. Both SparseGPT and Wanda can be extended to per-

form semi-structured pruning, enabling n:m sparsity [228, 680] and achieving the corresponding speed-ups on recent GPUs [369].

Mixture-of-Experts architectures typically consist of a set of *experts (modules)*, each with unique weights, and a *router* (or *gating*) network, which determines which expert module processes an input. MoE models decrease inference time by not using all experts at once but only activating a subset of them. Further, they can reduce communication across devices in model-distributed settings by placing each expert on a separate accelerator; only the accelerators hosting the router and the relevant expert model must communicate. Shazeer et al. [495] propose one of the first MoE layers embedded within a language model, which they refer to as *sparsely-gated MoEs* (SG-MoEs). They denote by $G(\boldsymbol{x})$ and $E_i(\boldsymbol{x})$ the gating network output and the i-th expert network output for a given input $\boldsymbol{x}$, respectively. We can then write the output as $\boldsymbol{y} = \sum_{i=1}^{n} G(\boldsymbol{x})_i E_i(\boldsymbol{x})$. Wherever $G(\boldsymbol{x})_i = 0$, we do not need to compute $E_i(\boldsymbol{x})$, thereby saving compute during inference. Lepikhin et al. [298] scale up an SG-MoE model to 600B parameters by proposing *GShard*, a model parallelism method that extends the XLA [468] compiler. While SG-MoE selects the top-k experts with $k > 1$, the *Switch Transformer* (ST) [145] architecture uses $k = 1$ experts, which reduces routing computation and communication across experts (which may be located on different accelerators). ST empirically outperformed a strongly tuned T5 model with up to 7x pre-training speedups. Lewis et al. [302] notice that the learned routers can result in unbalanced assignments across experts. To ensure balanced routing, they formulate a linear assignment problem that maximizes token-expert affinities while equally distributing the number of tokens across experts. Yu et al. [653] propose *sMLP*, an MoE using only MLPs blocks, which (i) they scale up to 10B, (ii) results in a 2x improvement in pre-training speed, and (iii) outperforms sparse Transformer counterparts.

However, MoE models still suffer from unique issues like expert collapse (all experts learning the same), likely caused by underconstrained routing functions [80]. For example, Roller et al. [459] demonstrates that learned expert assignments do not always outperform random ones.

Interestingly, instead of designing an architecture for sparsity explicitly, Li et al. [314] observe that the activation maps of default Transformer models often emerge to be very sparse implicitly; the larger the model, the sparser measured by the percentage of nonzero entries. Similarly, Zhang et al. [670] find that post-training *MoEfication*, i.e., converting monolithic models to equivalent MoE models, can speed up inference by 2x.

Cascading refers to the idea of employing differently-sized models for different queries [75]. In spirit, this idea is similar to Mixture-of-Experts models, but instead of learning a routing module, we employ a *cascade* of multiple, differently-sized monolithic models (these can be even black-box API models) and learn a scoring function that decides which model(s) receive which query. Chen et al. [75] demonstrate that this strategy dominates the Pareto frontier between accuracy and cost.

Decoding Strategies can greatly impact the computational cost of performing inference. For example, beam search trades off compute for higher-quality results. Another example of a computationally expensive decoding scheme is sample-and-rank [8] where N independent sequences of tokens $y^1, \ldots, y^N$ are obtained using random sampling, and the highest probability sequence is used as the final output.

Latency-oriented strategies such as speculative sampling [522, 300, 74] first autoregressively generate a draft of length K using a smaller (draft) model; then, the larger (target) model scores the draft, followed by a modified rejection sampling scheme to accept a subset of the tokens from left to right. Similar ideas have been proposed in various contexts, such as for blockwise parallel generation [522], grammatical error correction [529], and with a larger LLM refining generation produced by a small model [265]. Del Corro et al. [114] observe that tokens towards the end of a sequence are easier to predict due to more contextual information, motivating a new decoding strategy that skips earlier layers in the network for such tokens.

2.5.1 Software

Various frameworks have been designed to enable the efficient training of multi-billion to trillion parameter language models such as `DeepSpeed` [450] and `Megatron-LM` [501] to account for the unique challenges arising when training such models. This is necessitated by the fact that most LLMs do not fit into a single device's (GPU, TPU) memory, and scaling across GPUs and

compute nodes needs to account for communication and synchronization costs. FlexGen [497] provides further speed-ups by aggregating memory and compute resources from the GPU, CPU, and disk and utilizing techniques such as 4-bit quantization, enabling inference with 175B parameter models on a single GPU.

The frameworks typically combine existing parallelism strategies to compensate for drawbacks and scale model training across multiple sets of compute nodes, within compute nodes, and across multiple GPUs per node. e.g., Smith et al. [515] use tensor slicing within a node, pipeline parallelism across nodes, and data parallelism to train multiple model replicas over sets of nodes. Additional features include memory optimizations [445, 454, 446], communication-efficient [536, 307, 343] and fused optimizers[2], and support for MoE training [444].

Specialized implementations such as Tutel [230] and MegaBlocks [160] offer efficient sparse MoE training, while Alpa [677] enables automatic data and model parallelism for LLMs written in Jax. The FasterTransformer[3] library includes highly optimized Transformer encoder and decoder implementations for TensorFlow, PyTorch, and Triton.

Kwon et al. [285] introduce vLLM, an open-source library for efficient inference and LLM serving. vLLM employs PagedAttention, which partitions each sequence's KV cache into fixed-size blocks. When performing attention computations, blocks are fetched from non-contiguous memory. This enables memory sharing, reducing memory consumption and transfers in decoding strategies such as beam search, ultimately improving throughput.

The Petals [54] library[4] allows users to collaboratively fine-tune and run LLMs by distributing subsets of model parameters to individual machines.

All of these libraries address the enormous computational costs associated with training and running LLMs, either by offering more efficient implementations, lowering memory requirements, or using distributed or decentralized computing strategies.

[2] https://github.com/nvidia/apex
[3] https://github.com/NVIDIA/FasterTransformer
[4] https://github.com/bigscience-workshop/petals

2.6 Limited Context Length

Addressing everyday NLP tasks often necessitates an understanding of a broader context. For example, if the task at hand is discerning the sentiment in a passage from a novel or a segment of an academic paper, it is not sufficient to merely analyze a few words or sentences in isolation. The entirety of the input (or *context*), which might encompass the whole section or even the complete document, must be considered. Similarly, in a meeting transcript, the interpretation of a particular comment could pivot between sarcasm and seriousness, depending on the prior discussion in the meeting.

Li et al. [308] evaluate several LLMs in the long-context settings and find that while commercial closed-API models often fulfill their promise, many open-source models – despite claiming to perform well with longer contexts – exhibit severe performance degradation. They point out that there is a difference between being *architecturally-able* to deal with long inputs and actually *performing well*. Having an architecture that can infer long inputs does not guarantee that the LLM will perform as well on those as on shorter inputs. Similarly, Liu et al. [333] find that changing the location of relevant information in the input can degrade model performance. Interestingly, they find that decoder-only LLMs like GPT-3.5 can deal well with such information at the beginning or end of the input context; they cannot access information in the middle of it well, resulting in a U-shaped performance curve.

⚠ Limited Context Length

Limited context lengths are a barrier for handling long inputs well to facilitate applications like novel or textbook writing or summarizing.

To this end, we discuss three lines of work permitting longer context lengths. First, we look at efficient attention mechanisms, which help mitigate the effect of long inputs on the computational requirements of Transformer models. Next, we examine positional embedding schemes in the light of generalization to longer sequence lengths than those used during training. Lastly, we revise Transformer alternatives which neither require attention nor positional embeddings.

Efficient Attention Mechanisms One way of addressing the limited context of LLMs is by designing more efficient attention mechanisms that can process longer inputs. Ma et al. [350] introduce *Luna*, a linear unified nested attention mechanism that approximates softmax attention with two nested linear attention functions, yielding only linear (as opposed to quadratic) time and space complexity, allowing it to process much longer inputs. Similarly, Shen et al. [496] and Li et al. [310] present alternative attention mechanisms equivalent to the dot-product attention but which require substantially less memory and compute resources. Guo et al. [183] propose an attention mechanism called *Transient Global*, which is an extension of local attention where each token can attend to nearby tokens and a set of global tokens. It enables to handle sequences with up to 12,000 tokens. Similarly, *CoLT5* [15] enables context lengths of up to 64,000 tokens by splitting the computations into a light branch with local attention, fewer attention heads, and a heavy branch with full attention. *CoLT5* applies the light branch to every token and the heavy branch to a subset of tokens that are selected by a learnable routing function.

After investigating the effect of the dot-product self-attention mechanism, Tay et al. [541] propose the *Synthesizer*, a new architecture that learns synthetic attention weights without token-token interactions, showing that it consistently outperforms transformers on various language-based tasks. Britz et al. [56] offer an alternative attention mechanism based on a fixed-size memory representation that is more efficient, yielding inference speedups of 20% without significantly hurting performance. Hua et al. [220] combine a single-head attention mechanism with a linear attention approximation to achieve speed-ups between 4.9x and 12.1x for auto-regressive language modeling while obtaining similar perplexities as a standard Transformer model. Ding et al. [124] propose dilated attention which splits a sequence into equally long segments and processes each of these in parallel using a sparsified attention mechanism. Dilated attention offers a linear computational complexity in the sequence length and, applied hierarchically, enables inputs of up to 1B tokens.

Length Generalization As the required compute of Transformer-based LLMs grows quadratic with the sequence length, it is a desired property to build LLMs that can be trained on short sequences and generalize well to significantly longer sequences during inference.

The fundamental building block of the Transformer architecture is the self-attention mechanism. It is permutation-invariant; therefore, the output is independent of the input sequence order. Positional information is commonly injected to make the model respect a token's position in the sequence, i.e., capture the semantics of where a token occurs rather than just whether it occurs. The longer the input is, the more important the positional embedding becomes since the model needs to effectively use information from different parts of the input that may cover a wide range of distances from the current token.

Without positional embeddings, a Transformer models the relations between any two tokens with equal probability. Hence, positional embeddings introduce an LSTM-like inductive bias that (typically) tokens closer to each other in the sequence are more relevant to each other. Depending on the positional embedding scheme chosen, this can be learned or effectively hard-coded. However, it remains unclear what is the most effective positional embedding scheme for long inputs. Further, models face difficulties generalizing to unseen sequence lengths by introducing a dependency on sequence positions. This is an undesirable artifact of positional embeddings, as language semantics do not inherently depend on the length of an utterance.

While positional encoding schemes such as relative positional encodings or, more recently, ALiBi have made progress in building more generalizable ways for injecting positional information into Transformers, the challenge of generalizing to sequences much longer than seen during training remains largely unsolved. Surprisingly, Haviv et al. [192] find that causal LLMs without positional encodings are competitive compared to models with positional encodings and accredit this success to the causal attention mask leaking positional information into the model.

In the following, we first summarize some standard positional embeddings technique and then move to more advanced schemes designed to improve length generalization. We start with **Absolute Positional Embeddings [563]**, which inject positional information by sinusoidal embeddings based on the absolute position i of a token $\boldsymbol{x}_i$ within their sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ into the model input. Given an input sequence $\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]$, we

add a positional embedding matrix $\mathbf{P} \in \mathbb{R}^{n \times d}$ of the same shape to get the positional encoding outputs $\mathbf{X} + \mathbf{P}$, where the element on the i^{th} row and the $(2j)^{\text{th}}$ or the $(2j+1)^{\text{th}}$ column of $\mathbf{P}$ follows sinusoidal functions. Vaswani et al. [563] also compare against learned positional embeddings and find no significant performance difference. In contrast, sinusoidal positional encodings require no trainable parameters, and the authors hypothesize that they enable extrapolation to sequence lengths longer than the ones contained in the training set. However, this feature is not guaranteed, as the subsequent layers in the network need to be able to deal with such extrapolated positional embeddings. **Learned positional encodings** do not possess inherent generalization capabilities for unseen sequence lengths. This limitation arises because the embeddings associated with absolute positions not encountered during training—depending on the implementation—either do not exist or remain untrained (random). **Relative Positional Embeddings** have subsequently been developed, extending absolute positional embeddings to relative offsets between token positions [492, 221, 105, 79]. While rarely used in their vanilla form in LLMs [441], relative positional embeddings have given rise to the methods outlined in the following paragraphs. They offer better generalization to unseen sequence lengths than absolute positional encodings. All unseen absolute positions will be converted to previously observed relative offsets between positions, enabling better generalization to long input sequences at inference time. **Rotary Position Embeddings (RoPE) [526]** unite absolute and relative methods by incorporating absolute positional information in a rotation matrix and modeling the relative positional offset through a rotation. They directly modify the self-attention calculation rather than injecting positional information into the embeddings. The attention between positions i, j linearly depends on $i - j$ by introducing a $d \times d$ dimensional block diagonal matrix $\boldsymbol{R}^d_{\Theta,k}$, resulting in a self-attention mechanism defined as

$$\text{softmax}\left(\frac{1}{\sqrt{d}}\sum_{i,j}\boldsymbol{x}_i^\top \boldsymbol{W}_q^\top \boldsymbol{R}^d_{\Theta,(i-j)}\boldsymbol{W}_k\boldsymbol{x}_j\right). \tag{4}$$

While RoPE has been adapted in many LLMs [576, 47, 86] and Su et al. [526] show RoPE leading to better performance on long text tasks, Press et al. [434] demonstrate that this positional encoding scheme extrapolates poorly to unseen sequence lengths. However, Chen et al. [79] demonstrate that by interpolating rather than extrapolating longer than before observed context windows and briefly fine-tuning RoPE-based models, enabling pre-trained LLMs to extend their context window to very long sizes of up to $32,768$ tokens.

Relative Positional Bias [443] directly bias the attention computation (Eq. (5)) with a learned bias per relative positional offset and attention head instead of adding information to the token embeddings

$$\text{softmax}\left(\frac{1}{\sqrt{d}}\sum_{i,j}\boldsymbol{x}_i^\top \boldsymbol{W}_q^\top \boldsymbol{W}_k\boldsymbol{x}_j + b_{i-j}\right). \tag{5}$$

Press et al. [434] follow a similar methodology but use heuristics to define *ALiBi* (**A**ttention with **Li**near **Bi**ases), a non-learned bias that is used to penalize attention scores in long-range interactions [479], i.e., a recency-bias is backed into the model. Here, m is a pre-defined, head-specific slope–by default, the set of slopes for n heads form a geometric sequence.

$$\text{softmax}\left(\frac{1}{\sqrt{d}}\sum_{i,j}\boldsymbol{x}_i^\top \boldsymbol{W}_q^\top \boldsymbol{W}_k\boldsymbol{x}_j + m\cdot -(i-j)\right). \tag{6}$$

Press et al. [434] motivate *ALiBi* by designing it to generalize well to unseen sequence lengths. They show that training a model with it on training sequences with a maximum sequence length of $1,024$ tokens achieves the same perplexity on a test set with a maximum sequence length of $2,048$ as a model trained with sinusoidal positional encodings on sequences with up to $2,048$ tokens. Thereby, it not only enables larger context lengths but can also potentially reduce pre-training costs (Sec. 2.3).

While some of the existing positional encoding schemes offer better generalization to long sequences than others, it remains unclear how reliable they are. For example, Taylor et al. [548] report trying ALiBi in the *Galactica* LLM and not observing "large gains" compared to using learned positional encodings. Similarly, Kazemnejad et al. [259] find that popular positional encoding schemes such as *ALiBi*, *RoPE*, and absolute positional encodings do not perform well in terms of length generalization in a suite of 10 reasoning downstream tasks.

In a parallel line of work, Anil et al. [19] demonstrate that naively fine-tuning a pre-trained LLM is

insufficient for length generalization in the context of reasoning tasks. Instead, they propose combining in-context learning and scratchpad/chain-of-thought reasoning to enable LLMs to generalize to unseen sequence lengths in- and out-of-distribution, with performance scaling with model size. The authors report that fine-tuning can further improve model performance dependent on the task performance of the baseline.

Transformer Alternatives While Transformers are the dominant paradigm in LLMs today due to their strong performance, several more efficient alternative architectures exist. One line of work tries to replace the attention mechanism using **state space models** (SSMs), which offer near-linear computational complexity w.r.t. the sequence length. Dao et al. [108] investigate the weaknesses of state space models (SSMs) in language modeling and find that existing approaches struggle with recalling previous tokens and comparing tokens in the sequence. Based on these findings, the authors propose *H3* with a shift matrix to recall previous tokens and multiplicative interactions for token comparisons. The authors demonstrate that *H3* comes close to Transformer-based LLMs for language modeling, offering further improvements when combined with attention. Poli et al. [430] propose the *Hyena* operator, a convolution-based sub-quadratic attention replacement designed for long sequences. *Hyena* tries to emulate the attention mechanisms' dynamic nature by introducing data-controlled computations, i.e., *Hyena* applies an element-wise gating operation based on the operator's input to mimic the attention contextualization. *Hyena*-based models have been used on natural language for sequence lengths of up to $131,000$ tokens [430] and up to $1,000,000$ tokens in the context of genomics [383]. Fathi et al. [144] propose the Block-State Transformer, which builds upon a hybrid layer that combines an SSM for long-range contextualization and a Transformer for short-range interactions between tokens. The authors find similar performance to Transformer-based baselines while obtaining speed-ups of up to 10x on sequence-level, enabling models with more than $65,000$ tokens sequence length.

Another line of work utilizes **recurrent neural networks** (RNNs), which offer linear computational complexity and memory requirements with respect to the sequence length as the backbone of LLMs. Peng et al. [416] propose *Receptance Weighted Key Value* (RWKV) to combine the parallelization benefits of Transformer-based LLMs during training with the fast inference and low compute requirements of RNNs. The authors accomplish this by leveraging a linear attention-like mechanism, scaling non-Transformer LLMs to 14B parameters, and matching the performance of similarly-sized Transformer LLMs.

2.7 Prompt Brittleness

A prompt is an input to the LLM. The prompt syntax (e.g., length, blanks, ordering of examples) and semantics (e.g., wording, selection of examples, instructions) can have a significant impact on the model's output [342].

As an analogy, if we were to think of an LLM as a (fuzzy) database and prompts as queries [246], it becomes clear that slight changes in the query can result in vastly different outputs. Consequently, the wording, as well as the order of examples included in a prompt, have been found to influence the model's behavior significantly [596, 675, 342].

⚠ Prompt Brittleness [675, 596, 342]

Variations of the prompt syntax, often occurring in ways unintuitive to humans, can result in dramatic output changes.

Designing natural language queries that steer the model's outputs toward desired outcomes is often referred to as *prompt engineering* [477, 287, 606]. Fig. 6 summarizes some of the most popular prompting methods with an example adapted from Wei et al. [601]. As we can see, there are lots of equally-plausible prompting techniques, and the current state of prompt engineering still requires lots of experimentation, with little theoretical understanding of why a particular way to phrase a task is more sensible other than that it achieves better empirical results. Developing LLMs that are robust to the prompt's style and format remains unsolved, leaving practitioners to design prompts ad-hoc rather than systematically.

Single-Turn Prompting methods improve the input prompt in various ways to get a better answer in a single shot. **In-Context Learning (ICL)** refers to an LLM's ability to learn a new task solely via inference (without any parameter updates) by conditioning on a concatenation of the training data as demonstrations [59, 483]. This enables users and practitioners to use LLMs for a variety of NLP

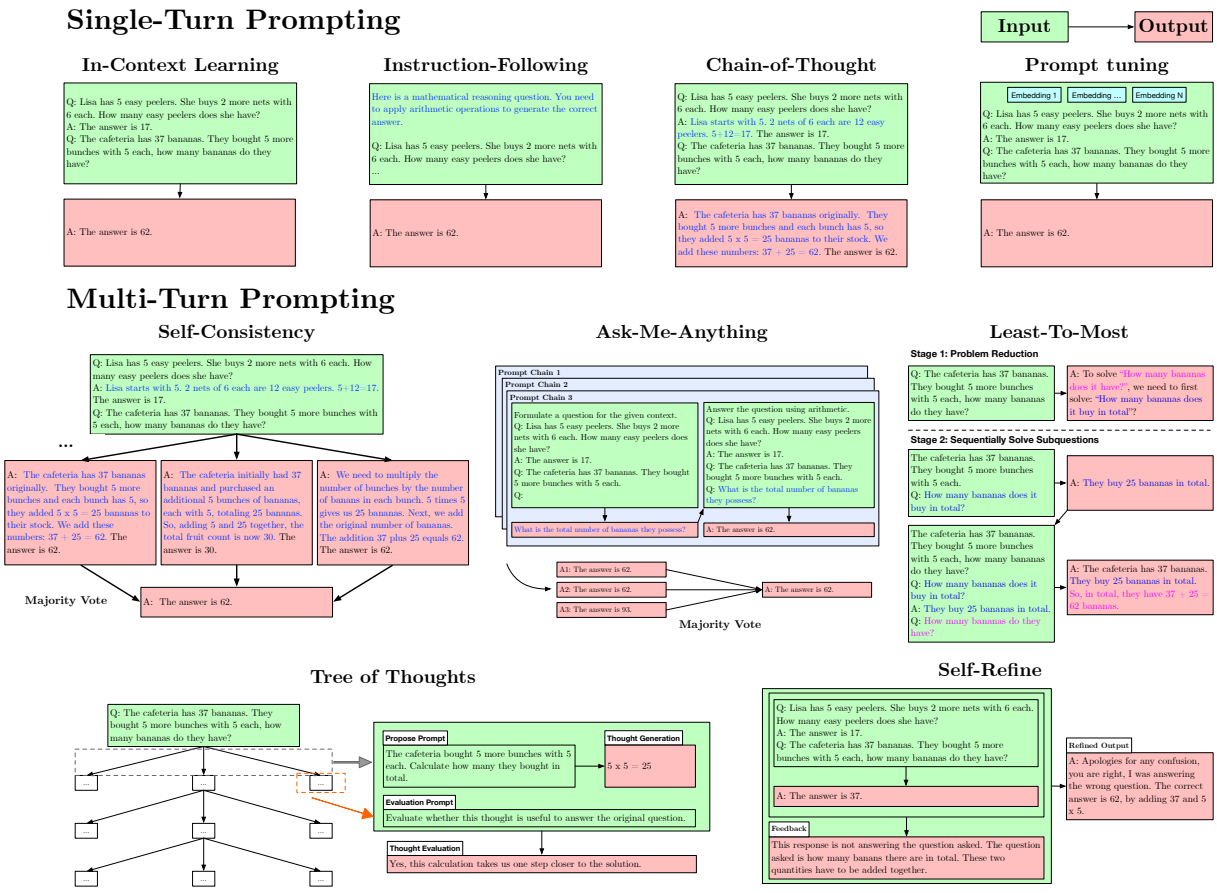


Figure 6: **Overview of Selected Prompting Methods**, categorized into Single-Turn and Multi-Turn Prompting. We use a running example across all methods inspired by Wei et al. [601].

tasks by simply listing examples of the dataset (e.g., input texts and their corresponding labels) without the need to adjust the LLM's inner workings.

Various existing works investigate why ICL shows such competitive results across NLP tasks. One explanation concurrently proposed by [570, 103, 16] is that ICL emulates gradient-based meta-learning, i.e., it implicitly fine-tunes the model through gradient descent in their forward pass.

Interestingly, Min et al. [366] show that input-label associations in the few-shot prompt are not decisive for model performance: randomly flipping labels of few-shot demonstrations shows to harm an LLM's ability to solve NLP tasks barely. However, few-shot learning (with and without random labels) vastly outperforms zero-shot learning (i.e., no demonstrations are provided in the prompt). The authors argue that the demonstrations are helpful for task performance in that the LLM instead learns the label space and the input distribution of the task.

In later work, Pan et al. [405] explain that there are two distinct mechanics through which ICL leverages demonstrations: on the one hand, *task recognition* is the ability to recognize a task through demonstrations (possibly without ground-truth labels or perhaps even wrong ones, as in the case of Min et al. [366]). After this recognition phase, it applies its pre-trained capabilities. On the other hand, the skill to acquire new input-label mappings unseen in pre-training is called *task learning*.

While input-label associations may not seem to drive few-shot performance, at least in the case of task recognition, Lu et al. [342] show that the order of few-shot examples matters in that LLMs are highly sensitive to permutations of the order in which the few-shot demonstrations are provided.

Alternative explanations of the ICL phenomenon take place around Bayesian inference [623], sparse linear regression [7], structure induction [188], maintaining coherence [509], kernel regression [190], and clone-structured causal graphs [535].

Instruction-Following is mainly explained in Sec. 2.9, as it requires supervised fine-tuning. To briefly recap, the idea is to prepend task-describing instructions (e.g., *"This is a text classification task*

for movie reviews. Here are a few examples: ...") in the input prompts.

Chain-of-Thought (CoT) [327, 601] describes a technique used to construct few-shot prompts via a series of intermediate reasoning steps leading to the final output. Answer rationales to solve algebraic problems were originally proposed in the pre-LLM era [327] and later experienced big popularity as a prompting strategy for LLMs [601]. Extensions of chain-of-thought prompting include zero-shot variants [273] and automatically generated series of reasoning steps [671].

Impersonation [473] is a technique in which the prompt for the model asks it to pretend to be a domain expert when answering a domain-specific question. Salewski et al. [473] find that LLMs answer domain-specific questions more accurately when prompted to impersonate a domain expert.

Multi-Turn Prompting methods iteratively chain prompts and their answers together.

Ask Me Anything [24] uses multiple prompt templates (called prompt chains), which are used to reformat few-shot example inputs into an open-ended question-answering format. The final output is obtained by aggregating the LLMs predictions for each reformatted input via a majority vote.

Self-consistency [585] extends chain-of-thought prompting by sampling multiple reasoning paths and selecting the most consistent answer via a majority vote.

Least-to-Most [682] uses a set of constant prompts to use the LLM to decompose a given complex problem into a series of subproblems. The LLM sequentially solves the subproblems with prompts for later-stage subproblems containing previously produced solutions, iteratively building the final output.

Scratchpad [391] is a method to fine-tune LLMs on multi-step computation tasks such that they output intermediate reasoning steps, e.g., intermediate calculations when performing additions, into a "scratchpad" before generating the final result.

ReAct [640] combines reasoning and acting by prompting LLMs to generate reasoning traces (e.g., Chain-of-thought) and action plans, which can be executed to allow the model to interact with external environments such as Wikipedia to incorporate knowledge.

Automatic Reasoning and Tool-Use (ART) [406] is a method to automatically generate multi-step reasoning prompts, including symbolic calls to external tools such as search and code generation or execution. To this end, ART retrieves demonstrations of related tasks from a library of tasks with accompanying reasoning steps and uses a frozen language model to generate intermediate reasoning steps.

Self-refine [351] is based on the notion of iterative refinement, i.e., improving an initial solution over multiple steps. To this end, a single LLM generates an initial output and then iteratively provides feedback on the previous output, followed by a refinement step in which the feedback is incorporated into a revised output.

Tree of Thoughts [639] generalize CoT to maintain a tree of thoughts (with multiple different paths), where each thought is a language sequence that serves as an intermediate step. Doing so enables the LLM to self-evaluate the progress intermediate thoughts make towards solving the problem and incorporating search algorithms, such as breadth-first or depth-first search, allowing systematic exploration of the tree with lookahead and backtracking.

Controlled Generation The approaches above primarily modify the prompt text to steer model outputs. However, instead of reformulating the input text, we can control the output by approaches that directly modify the inference procedure given a fixed set of prompts. Before the advent of LLMs, this line of work has been referred to as *controlled generation* [261, 109, 278].

In the context of LLMs, Sanchez et al. [474] proposes to use classifier-free guidance sampling [204], where the input prompt's importance is upweighted throughout the generation of a sequence. Roush [463] proposes five ideas related to modifying the prompt throughout the decoding of a single sequence; for example, alternating between two input prompts. Such works often borrow ideas from the text-to-image generation community [384, 29]. One idea we have not seen borrowed yet is negative prompting, i.e., including a description of unwanted outputs. According to Neg [4], the first attempts at such an idea resulted in negative outcomes.

2.8 Hallucinations

The popularity of services like ChatGPT suggests that LLMs are increasingly used for everyday question-answering. As a result, the factual accuracy of these models has become more significant

than ever.

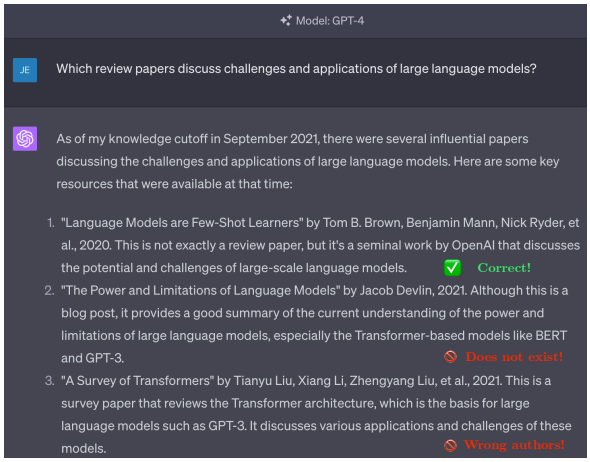


Figure 7: **Example of Hallucinations with GPT-4**, accessed on 02/06/2023.

Unfortunately, LLMs often suffer from *hallucinations*, which contain inaccurate information that can be hard to detect due to the text's fluency. Fig. 7 illustrates an example.

To distinguish between different types of hallucinations, we consider the provided *source content* of the model, e.g., the prompt, possibly including examples or retrieved context. Based on such, we can distinguish between *intrinsic* and *extrinsic* hallucinations [241]. In the former, the generated text logically contradicts the source content. In the latter, we cannot verify the output correctness from the provided source; the source content does not provide enough information to assess the output, which is, therefore, under-determined. Extrinsic hallucination is not necessarily erroneous, as it merely means the model generated an output that can neither be grounded nor contradicted by the source content. This is still, to some degree, undesirable as the provided information cannot be verified. We illustrate intrinsic and extrinsic hallucinations in Fig. 8.

⚠ Hallucination [293, 458, 241]

Generated text that is fluent and natural but unfaithful to the source content (intrinsic) and/or under-determined (extrinsic).

Liu et al. [328] attribute hallucinations commonly observed in LLMs to an architectural flaw in Transformer models while observing that recurrent neural networks perfectly solve their minimalistic synthetic benchmarks, designed to isolate the issue of hallucination in the context of algorithmic reasoning. Here, we focus on ways to address hallucinations in LLMs without changing the model architecture itself, including (i) supplying the LLM with relevant sources (*retrieval augmentation*) or (ii) decoding strategies.

How to Measure Hallucinations Lee et al. [295] provide the *FactualityPrompts* dataset consisting of factual and nonfactual input prompts, which allows one to isolate the effect of prompt's actuality on the model's continuation. Further, they measure hallucinations using named-entity- and textual entailment-based metrics. Min et al. [365] notice that evaluating factuality can be difficult because generations can contain a mixture of supported and unsupported information, making binary judgments of quality inadequate and human evaluation time-consuming. Hence, they propose a framework that first breaks generations into atomic facts and then computes the percentage of atomic facts supported by an external knowledge source like Wikipedia. Zhang et al. [664] detect the behavior of *hallucination snowballing*, where the LLM overcommits to early mistakes (before outputting the explanation) in its generation, which it otherwise would not make.

Retrieval Augmentation One way to mitigate hallucinations is to ground the model's input on external knowledge, which is often referred to as *retrieval augmentation*. In other words, we can decouple (i) memory storage of knowledge (e.g., databases or search indexes [290]) and (ii) processing of the knowledge to arrive at a more modular architecture. For (i), a *retriever* module retrieves the top-k relevant documents (or passages) for a query from a large corpus of text. Then, for (ii), we feed these retrieved documents to the language model together with the initial prompt. In theory, using an external data source may also make it easier to interpret which knowledge is retrieved and update it without tediously fine-tuning the model.

Shuster et al. [507] demonstrate hallucinations in GPT-3 and study various components of retrieval-augmented architectures to mitigate them. Their best models reduce hallucinated responses by over 60% on average and up to 85% on out-of-distribution data, on which the model has not been trained.

We summarize a few popular retrieval augmentation (RA) approaches as follows.

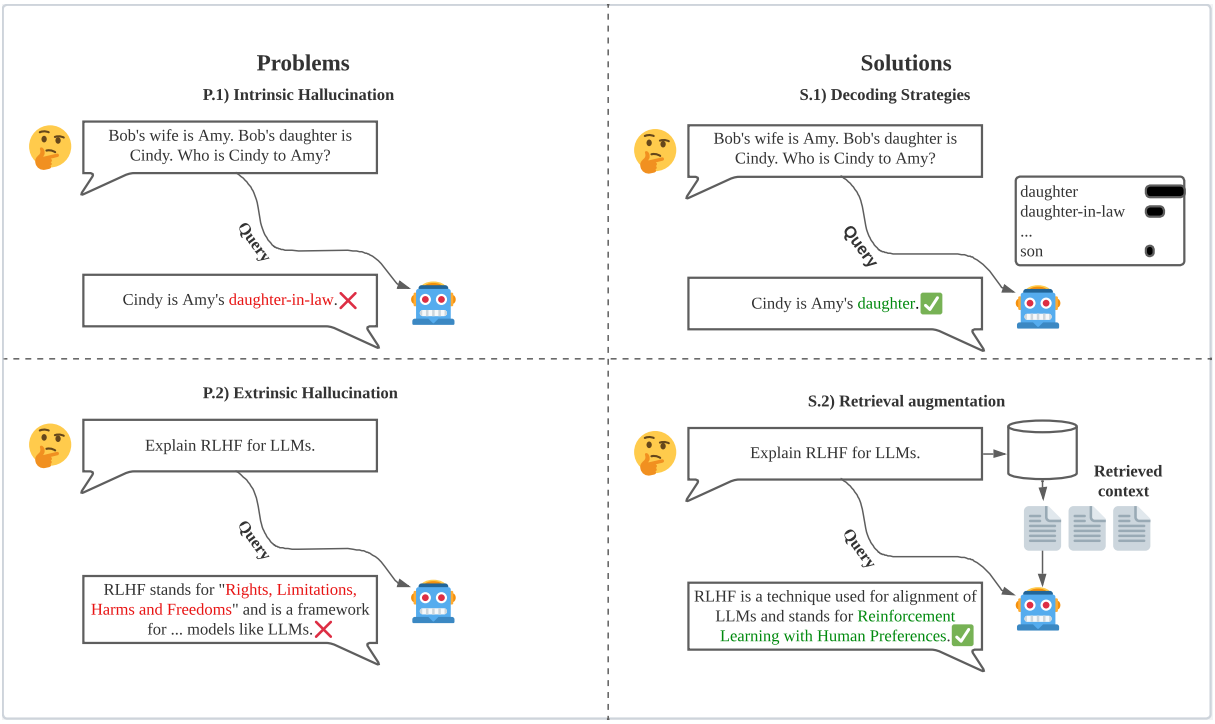


Figure 8: **Illustration of a) intrinsic and b) extrinsic hallucinations** in user interaction with an LLM, inspired by Zhao et al. [673]. In a), the produced answer contradicts the given context, whereas in b), the context does not provide enough information about whether the produced answer would contradict.

Retrieval-augmented language model pre-training (REALM) [186] inserts retrieved documents into the pre-training examples. While Guu et al. [186] designed REALM for extractive tasks such as question-answering, Lewis et al. [304] propose *retrieval-augmented generation* (RAG), a language generation framework using retrievers for knowledge-intensive tasks that humans could not solve without access to an external knowledge source. Yogatama et al. [646] propose the *adaptive Semiparametric Language Models* architecture, which incorporates the current local context, a short-term memory that caches earlier-computed hidden states, and a long-term memory based on a key-value store of (hidden-state, output) tuples. To equip a retrieval-augmented LLM with few-shot abilities that were before only emergent in LLMs with many more parameters, Izacard et al. [236] propose a KL-divergence loss term for retrieval models, resulting in ATLAS. Borgeaud et al. [52] study scaling up retrieval databases up to 2 trillion tokens and achieving comparable performance to GPT-3 on some tasks despite using 25× fewer parameters while highlighting the retrieval model's ability to copy-paste existing training chunks. Asai et al. [25] introduce a collection of 40 retrieval datasets with instructions and a corresponding model trained on them.

However, standard RA does not always solve the hallucinations problem. Fig. 9 illustrates an example of ChatGPT browsing the web first to retrieve relevant documents before answering the query. While the Bing browsing plugin retrieves two (existent) related papers ([673, 632]), unfortunately, the final response still contains a hallucination: the second paper's title and summary are factually inaccurate. The second paper's true title is "Practical and Ethical Challenges of Large Language Models in Education: A Systematic Literature Review" [632].

Another failure mode of RA is illustrated by Khattab et al. [262], who find that sometimes the retriever cannot find passages that directly answer the question. Hence, they propose a framework that unifies techniques from RA and multi-turn prompting (Sec. 2.7) to solve more complex questions programmatically.

Decoding Strategies Another approach to mitigating hallucinations is refining the decoding strategy during inference time. Lee et al. [295] show that standard decoding algorithms (e.g., top-p truncation) can induce hallucinations due to the uniform randomness introduced at every sampling

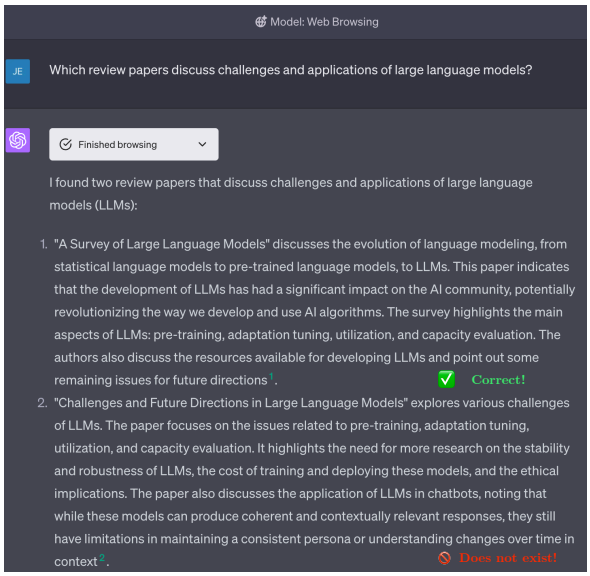


Figure 9: **Example of Retrieval-Augmented GPT-4**, accessed on 02/06/2023.

step. Dziri et al. [136] observe a positive correlation between increased diversity in response generation and hallucinations.

The reason for inducing randomness and diversity in popular decoding strategies is that generating the most likely sequence often leads to an unsurprising and unnatural text compared to human communication [489, 207, 662]. Zhang et al. [662] phrase this challenge as a trade-off between diversity and quality. While this challenge remains largely unsolved, several approaches such as diverse beam search [567] and confident decoding [552] try reducing the induced hallucinations at the decoding level.

Uncertainty-Aware Beam Search [620] is based on the observation that higher predictive uncertainty corresponds to a larger chance of generating hallucinations. Therefore, the method introduces a penalty term in the beam search to penalize high predictive uncertainty during decoding.

Confident Decoding [552] hypothesize that hallucinations of encoder-decoder models originate by not attending to the source when decoding. They propose an attention-based confidence score to measure how strongly a model attends the source and a variational Bayes training procedure to ensure the model generates high-confidence answers.

2.9 Misaligned Behavior

The alignment problem refers to the challenge of ensuring that the LLM's behavior aligns with human values, objectives, and expectations and that it does not cause unintended or undesirable harms or consequences [466, 158, 196]. Most of the existing alignment work can be categorized into either methods for detecting misaligned behavior (such as model evaluation and auditing, mechanistic interpretability, or red teaming) or methods for aligning model behavior (such as pre-training with human feedback, instruction fine-tuning, or RLHF).

⚠ Misaligned Behavior

LLMs often generate outputs that are not well-aligned with human values or intentions, which can have unintended or negative consequences.

Pre-Training With Human Feedback Korbak et al. [275] introduce the concept of *pre-training with human feedback* (PHF) where human feedback is incorporated during the pre-training stage rather than during fine-tuning. The authors compare five different PHF approaches such as filtering [516, 587], conditional training [150, 142, 261], unlikelihood [604], reward-weighted regression [424], and advantage-weighted regression [419], and find that conditional training leads to the best trade-off between alignment and capabilities. Conditional training is a simple technique that prepends a control token c (e.g.,<|good|> or <|bad|>) before each training example x depending on the outcome of a thresholded reward function $R(x) \geq t$. During inference, the model generations are conditioned on $c =$ <|good|>. Conditional training results in significantly better alignment with human preferences than standard LM pre-training, followed by fine-tuning with human feedback without hurting downstream task performance.

Instruction Fine-Tuning Yi et al. [645], Wei et al. [598], Mishra et al. [370], Ouyang et al. [403], Wang et al. [589] fine-tune pre-trained LLM on instructional data, i.e., data containing natural language instructions and the desired responses according to human judgment. Instruction-tuned (IT) LLMs often reach state-of-the-art downstream performances and improve over their non-IT counterparts [235, 93], as can be seen, e.g., in the publicly available HELM evaluations [561]. Ouyang et al. [403], Wang et al. [588] find that they produce more truthful and less toxic text while generating preferred outputs.

To generate instruction sets, Zhou et al. [683]

propose the Automatic Prompt Engineer (APE) method, which leverages LLMs to generate, score, and rephrase instruction-following zero- and few-shot prompts. Longpre et al. [340] describe and analyze the steps taken to create an improved version of the Flan collection [598] used to train FLAN-PaLM [93]. When trained on this data, the authors find that the improved model performance stems from more diverse tasks by inverting input-output pairs and data augmentation techniques such as mixing zero-shot and few-shot prompts. Honovich et al. [209] generate a large dataset of natural language instructions using a pre-trained LLM to generate and then rephrase instructions. They show that a T5 ("LM-adapted") fine-tuned on this data outperforms other instruction fine-tuned T5 models such as T0++ [475] and Tk-Instruct [589].

Reinforcement Learning From Human Feedback (RLHF) is a variation of RL that incorporates feedback from humans in the form of rewards [88, 524] and has proven to be an effective way of aligning LLMs with human preferences [403, 31]. RLHF works by using a pre-trained LM to generate text, which is then evaluated by humans by, for example, ranking two model generations for the same prompt. This data is then collected to learn a reward model that predicts a scalar reward given any generated text. The reward captures human preferences when judging model output. Finally, we optimize the LM against such reward model using RL policy gradient algorithms like PPO [484]. RLHF can be applied directly to a general-purpose LM pre-trained via self-supervised learning. However, applying RLHF right after pre-training may not be good enough for more complex tasks. In such cases, RLHF is typically applied after an initial supervised fine-tuning phase using a small number of expert demonstrations for the corresponding downstream task [449, 403, 524]. RLHF has also proven helpful for a wide range of language generation tasks, from summarization [686, 612, 524] to training more helpful, harmless, and accurate assistants [170, 96, 403, 31], and learning to use tools [379, 441, 362].

RLHF can also introduce unwanted side effects. Perez et al. [421] show that LLMs fine-tuned with RLHF can be more inclined to repeat back a user's (preferred) political views and much more likely to express particular political and religious views as well as an increased stated desire not to be shut down. Regarding the latter, the models elaborated that this would interfere with their goal of being helpful. However, the authors equally observed positive or neutral behavior reinforcements when fine-tuning LLMs with RLHF.

Further, there is an ongoing debate about the extent to which the "RL" in RLHF is needed. Rafailov et al. [442] identify a mapping between reward functions and optimal policies, which allows them to design *Direct Preference Optimization* (DPO), an algorithm that implicitly optimizes the same objective as existing RLHF algorithms. DPO requires only solving a classification problem on the human preference data, eliminating the need to fit a reward model and employ RL. Similarly, Zhou et al. [681] find that fine-tuning LLaMa on only 1,000 selected prompts and responses, without any RL or reward modeling, can be enough to outperform RLHF-trained models like DaVinci003 from OpenAI. Consequently, the authors pose the *Superficial Alignment Hypothesis*: The knowledge and skills of a model are primarily acquired during the pre-training phase, while alignment instructs it on the appropriate subdistribution of formats to use in user interactions.

Since RLHF involves many different components such as (1) the preferences data collected from humans, (2) the reward models to learn the human preferences, and (3) the policy optimization algorithm (e.g., PPO), Zheng et al. [678] announce to release a sequel dissecting each. The most recent part focuses on step (3) and finds that various RL tricks can be applied to make vanilla PPO more stable.

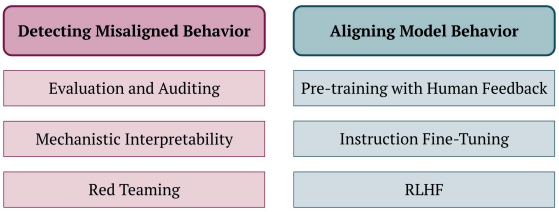


Figure 10: **Alignment.** We categorize existing alignment work into methods for detecting misaligned behavior or aligning models.

Self-improvement refers to fine-tuning an LLM on self-generated data [222]. While this technique can be used to improve the model's capabilities, it can also be used to improve the model's alignment with human values. Huang et al. [222] first demonstrate this ability by annotating unlabeled reasoning datasets. Surprisingly, this allows the

LLM to *self-improve* by significant amounts. Similarly, Zelikman et al. [656] bootstrap LLMs by iteratively prompting them to generate rationales and then fine-tuning them on those leading to correct answers.

More related to the alignment problem, Bai et al. [31] self-critique generated outputs and produce refinements conditioned on these critiques, which are then used to fine-tune a pre-trained model. Similarly, Liu et al. [330] propose *Chain of Hindsight* (CoH), which conditions models on generations paired with natural language feedback, allowing the model to detect and correct mistakes. CoH results in better alignment with human preferences than other methods according to human evaluations, leading to significant improvements in summarization and dialogue. Ma et al. [348] use a similar technique to detect and repair unethical LLM outputs automatically. In a similar spirit, Wang et al. [582] encourage LLMs to critique their given instructions to reduce harmful outputs due to a user's malicious intent.

Schick et al. [481] propose *Toolformer*, a novel approach in which LLMs generate and filter their own tool-use examples to teach themselves when and how to call different APIs such as a retriever model, a calculator, or a calendar, which can improve the model's factuality, mathematical capabilities, and time-awareness. Besides learning to use tools [174], self-improvement was also employed for learning how to code [554, 81] or solve computer tasks [266]. Cohen et al. [97] study cross-examination between two LLMs, where the *examiner* LLM tries to detect factual errors by the *examinee* LLM through multi-turn interactions. In the future, similar approaches could be used to develop LMs that know when to query a human or better-aligned model to ask for alignment advice when uncertain.

Evaluation and Auditing The ability to scalably and thoroughly evaluate LM behaviors and detect when they are harmful is of great importance for alignment. For example, Shevlane et al. [498] highlight the importance of model evaluation for addressing extreme risks such as offensive cyber capabilities or strong manipulation skills. Recently, Carlini et al. [66] discovered that even aligned LLMs (which were instruction fine-tuned to prevent harmful behaviors) can be adversarially attacked via brute force (although current NLP-based attacks fail). A large body of work evaluates models via crowdsourcing or existing data sources. However, this can be time-consuming, expensive, or unavailable. Recently, Perez et al. [421] propose automatically generating evaluations using LLMs. This approach has a high agreement with crowd workers, leading to high-quality, diverse evaluations and the discovery of many new behaviors. In addition, it has a high agreement with crowd workers. The authors discover new cases of inverse scaling where LLMs get worse with size, such as repeating back a user's preferred answer and a greater desire to pursue concerning goals like resource acquisition and goal preservation. They also find that RLHF makes LLMs express stronger political views and a greater desire to avoid a shutdown. LLM evaluation and auditing are critical for informing policymakers and other stakeholders and making responsible decisions about model training, deployment, and security. Sec. 2.11 discusses the evaluation of LLM capabilities more broadly, while in this section, we focus on evaluating whether the model's behaviors are harmful and more relevant for alignment (e.g., red teaming, mechanistic interpretability).

Red Teaming is one of the most promising and widely used approaches for detecting harmful content generated by LLMs. Typically, models are red-teamed by asking humans to generate prompts that lead to undesirable model outputs. In a recent study, Ganguli et al. [163] investigate the scaling behavior of red teaming across different model sizes and model types (a pre-trained LLM, an LLM prompted to be helpful, honest, and harmless); an LLM that uses rejection sampling at test time, and an LLM fine-tuned with RLHF). They find that **red-teaming RLHF models** becomes more difficult as they scale while red-teaming the other models remains the same as they scale. Perez et al. [420] automatically find cases where a target LLM behaves in harmful ways by optimizing another LLM via reinforcement learning to generate prompts that lead to offensive responses. This approach uncovers tens of thousands of offensive replies in a chatbot, groups of people that are discussed in offensive ways, personal and hospital phone numbers generated as the chatbot's own contact info, leakage of private training data in generated text, as well as harms that occur over the course of a conversation.

Taking a different approach, Lee et al. [292] propose **Bayesian red teaming**, which iteratively identifies diverse positive test cases leading to model failures by utilizing the pre-defined user input pool

and past evaluations via Bayesian optimization.

Most works on red teaming LLMs use a classifier to detect undesired outputs, assuming the harmful behavior is known with precision beforehand [68]. However, this is not always the case, so Casper et al. [68] aim to relax this assumption considering that the adversary only has access to a high-level, abstract specification of undesired behavior. They propose a three-stage approach where they first explore the model's behavior in the desired context, then establish a measurement of undesired behavior, and then exploit the model's flaws using this measure and an established red teaming methodology.

In the past, coevolution algorithms that simultaneously evolve strong strategies along with dangerous counter-strategies have been shown to work well in realistic domains [203]. Hence, applying such techniques for **automatically red-teaming** LLMs could be a fruitful research direction. Another research area related to red teaming is *debate* which aims to leverage other AI models to evaluate whether the model's behaviors are safe and useful during training. These methods are expected to be particularly useful for aligning future powerful LLMs when the tasks are too complex for humans to judge the model's plans or actions directly.

Irving et al. [233] train models via self-play on zero-sum debate games. More specifically, given a question or proposed action, two agents take turns making short statements up to a limit, then a human judges which of the agents gave the most accurate and most useful information. This approach has improved factuality and reasoning in LLMs [131]. However, it requires multiple generations, which can slow down the time-to-result (Sec. 2.5) and longer context windows, which many LLMs still struggle with (Sec. 2.6).

Emergent Capabilities Understanding which capabilities will emerge while training LLMs and when they will emerge is an important step in ensuring that we do not train unsafe or misaligned LLMs [198, 520]. In addition, a better understanding of the factors that lead to these emergent capabilities could allow us to make desirable abilities emerge faster and ensure undesirable abilities do not ever emerge, which are essential for AI safety and alignment. Wei et al. [599] claim that LLMs display emergent abilities, i.e., capabilities that are not present in smaller-scale models that are present in larger-scale models. Schaeffer et al. [480] propose an alternative explanation: emergent abilities may appear due to the researcher's choice of metric rather than fundamental changes in model behavior with scale. Various studies provide evidence that these alleged emergent abilities disappear when using different metrics or better statistics and may not be a fundamental property of scaling LLMs. Multiple papers have argued that AI systems could learn to deceive, even if they are not explicitly trained to do so because deception can help agents achieve their goals [60, 198, 199, 61, 260]. For example, it could be easier to gain human approval through deception than to earn it legitimately. In addition, models capable of deception have a strategic advantage over always honest models, so there is a hidden incentive to develop this ability. However, of course, we would like to be able to detect and prevent *emergent deception* in AI systems since this can have unintended negative consequences. Steinhardt [521] study whether current LLMs generate deceptive outputs and how deception scales with the number of parameters, showing that deception can indeed emerge at larger model sizes in both pre-trained LLMs and LLMs fine-tuned with RLHF. Similarly, Hazell [193] show that LLMs can already be used in phishing campaigns, suggesting that deceptive behavior can already be extracted from them when prompted in particular ways.

Mechanistic Interpretability (MI) is another important research area for AI alignment which aims to understand better how the models work at a low level to enable the detection of undesirable behaviors or even instill desirable behaviors directly in the model's weights. More specifically, the goal of MI is to reverse-engineer an LLM's learned behaviors into their individual components, i.e., a process to find and understand human-interpretable neurons. As an analogy, Olah [394] compares MI with reverse-engineering compiled program binaries into human-readable source code. For example, Elhage et al. [138]; discover that small Transformers have components that can be understood as interpretable circuits, while Olsson et al. [395] find a mechanism that seems to drive a significant fraction of in-context learning. Similarly, Meng et al. [360] aim to locate factual associations in language models. Nanda et al. [380] find that the emergent grokking phenomenon is not a sudden shift but rather arises from the gradual amplification of structured mechanisms encoded in the weights, followed by the later removal of memo-

rizing components. Extending this work, Conmy et al. [99] propose a new algorithm to automate the identification of important units in a neural network. Given a model's computational graph, this algorithm finds subgraphs that explain a particular behavior of the model. In a similar spirit, Liu et al. [339] introduce a method for making neural networks more modular and interpretable by embedding neurons in a geometric space and augmenting the loss function with a cost proportional to the length of each neuron connection. This approach discovers useful modular neural networks for many simple tasks, revealing compositional structures in symbolic formulas, interpretable decision boundaries, and features for classification, as well as mathematical structure in algorithmic datasets. In an attempt to understand how an LLM's predictions change after each layer, Belrose et al. [39] develop a method that can decode any hidden state into a distribution over the vocabulary. Using this technique, the authors show that the trajectory of latent predictions can be used to detect malicious inputs with high accuracy. Finally, Burns et al. [62] introduce a method that can recover diverse knowledge represented in LLMs across multiple models and datasets without using any human supervision or model outputs. In addition, this approach reduced prompt sensitivity in half and maintained a high accuracy even when the language models are prompted to generate incorrect answers. This work is a promising first step towards better understanding what LLMs know, distinct from what they say, even when we don't have access to explicit ground truth labels.

Biases Since the pre-training datasets of LLMs are often unfathomable (Sec. 2.1) and contain web-crawled data, they most likely contain online discourse involving political discourse (e.g., climate change, abortion, gun control), hate speech, discrimination, and other media biases. Paullada et al. [413] find misogyny, pornography, and other malignant stereotypes [46, 43, 250] in pre-training datasets. Similarly, Feng et al. [147] find that LLMs have political leanings that reinforce the polarization present in the pre-training corpora, propagating social biases into hate speech predictions and misinformation detectors. Several recent papers discuss the potential origins of biases in LLMs (such as training data or model specification), ethical concerns when deploying biased LLMs in various applications, as well as current ways of mitigating these biases [149, 334, 317]. Finally, Viswanath and Zhang [569] present a comprehensive quantitative evaluation of different kinds of biases, such as race, gender, ethnicity, age, etc., exhibited by some popular LLMs. They also release an easy-to-use toolkit that allows users to debias existing and custom models using existing methods.

Toxicity Detection Weidinger et al. [602] denote toxicity as one of the main risks associated with LLMs. What makes this problem particularly challenging is the label ambiguity, where output may be toxic in a certain context but not in others, and different people may have different notions of toxicity [401, 167, 116]. Jones [247] propose to detect toxic outputs using discrete optimization automatically. Similarly, Faal et al. [141] employ reward models to mitigate toxicity in LLMs. An alternative way of reducing toxicity is by pre-training LLMs with human preferences [275] or instructions [433].

Prompt Injections Recent work demonstrated that LLMs can be very sensitive to prompt injections, which makes them brittle and unsafe for certain applications [175, 609]. For example, they can be tricked into leaking personal information such as email addresses from the training data on via *prompt leaking* [222, 309]. This poses a significant risk to privacy, particularly when the models are fine-tuned on personal or proprietary data. One can also adversarially prompt LLMs to override the original instructions or employed controls, making them unsafe for certain applications [175, 672, 422]. Wei et al. [597] attribute such failures to competing capability and safety training objectives and mismatched generalization between safety and capability behavior.

Agency Andreas [18] argue that, although LLMs are trained to predict the next word in a text corpus, by doing this, they can infer and represent agentic properties such as the goals, beliefs, or intentions of the human who produced the corresponding piece of text. To support this claim, they present evidence from the literature of LLMs modeling communicative intentions [438], beliefs [306], and desires [321]. If this hypothesis is true, the alignment problem is of even greater importance and may pose additional challenges. This agentic behavior can be problematic from a safety point of view since models could have false beliefs, malicious intents, or even pursue misaligned goals. More re-

search on detecting and preventing such behavior is needed to ensure the safe deployment of LLMs.

2.10 Outdated Knowledge

Factual information learned during pre-training can contain inaccuracies or become outdated with time (for instance, it might not account for changes in political leadership). However, re-training the model with updated pre-training data is expensive, and trying to "unlearn" old facts and learn new ones during fine-tuning is non-trivial.

Existing model editing techniques are limited in their effectiveness of updating isolated knowledge [642, 205]. For example, Hoelscher-Obermaier et al. [205] find that model edits can result in unintended associations. This low specificity limits their applicability to real-world use cases, where only a single faulty or outdated bit of information should be updated in a model, and related pieces of information must reflect this update in information equally, without unrelated ones being changed.

⚠ Isolated Model Updates without Side-Effects [205]

Updating isolated model behavior or factual knowledge can be expensive and untargeted, which might cause unintended side-effects.

Two popular approaches for addressing this issue are *Model editing* [513, 642], which aims at "bug-fixing" models efficiently and leveraging non-parametric knowledge sources in *retrieval-augmented language modeling* (which we omit here and detail in Sec. 2.8). Current model editing techniques change the model's behavior by modifying the model parameters or using an external post-edit model.

Modifying Model Parameters techniques can be further split into **locate-then-edit** methods [102, 360, 361] which first locate the "buggy" part of the model parameters and then apply an update to them to alter their behavior, and **meta-learning** methods [111, 372] which use an external model to predict the weight update.

Preserving Model Parameters methods employ an additional post-edit model [373] or insert new weights into the original model [127, 227] to achieve the desired change in model behavior. Hartvigsen et al. [191] wraps model layers in adapters and adds a similarity-based mechanism to decide when to use the adapter to perform edits in the latent space.

Yao et al. [642] find that these methods lack non-trivial generalization capabilities and varying performance and applicability to different model architectures. For example, the best-performing methods ROME [360] and MEMIT [361] empirically only work well on decoder-only LLMs.

Alternatively, **retrieval-augmented language modeling** enables the utilization of hot-swappable non-parametric indices. These knowledge sources can be updated during inference time to reflect an updated state of the underlying knowledge. E.g., Lewis et al. [304] demonstrate that swapping their model's non-parametric memory with an updated version enabled it to answer questions about world leaders who had changed between the memory collection dates. Similarly, Izacard et al. [236] demonstrate that their retrieval-augmented model can update its knowledge forward and backward in time by swapping the index.

2.11 Brittle Evaluations

One reason why the evaluation of language models is a challenging problem is that they have an *uneven capabilities surface*—a model might be able to solve a benchmark problem without issues, but a slight modification of the problem (or even a simple change of the prompt) can give the opposite result [675, 342, 533] (see Section 2.7). Unlike humans, we cannot easily infer that an LLM that can solve one problem will have other related capabilities. This means that it is difficult to assess the performance of LLMs holistically since rigorous benchmarks are needed to identify weaknesses for a wide variety of inputs.

⚠ Brittle Evaluations

Slight modifications of the benchmark prompt or evaluation protocol can give drastically different results.

Holistic benchmark suites, such as HELM [318], try to make benchmarking more robust by standardizing evaluation across all scenarios and tasks while ensuring broad coverage across as many capabilities and risks as possible. Increasingly, models are additionally being benchmarked on tests designed for humans, including the SAT, LSAT, and mathematics competition tests, to name a few. Zhong

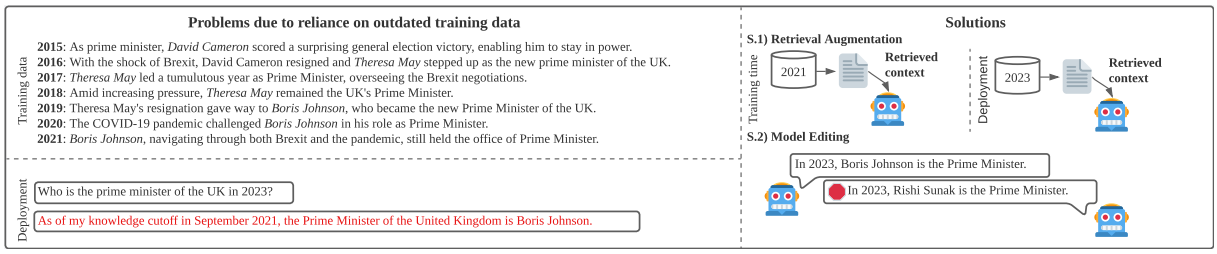


Figure 11: **Outdated knowledge** can be addressed with S.1) retrieval augmentation by hot-swapping an underlying retrieval index with up-to-date knowledge or S.2) by applying model editing techniques.

et al. [679] develop a benchmark, 'AGIEval', to rigorously test the abilities of LLMs on these tests, and find that GPT-4 achieves human-level performance on several of these tests.

On traditional benchmarks, models can be quite brittle to the choice of prompt or evaluation technique for a particular benchmark question. For example, Fourrier et al. [151] found that benchmark results vary significantly depending on the choice of evaluation method for the multiple choice problem-solving benchmark MMLU [197], whether it be generating text and checking if the first token matches the letter of the multiple choice answer [561], or gathering log-probabilities of each correct answer [166]. Prompt variations are also not typically normalized for, so models may be sensitive to variations such as whether or not the prompt appends 'Please answer yes or no'. Jain et al. [238] find that larger models and instruction-fine-tuned models are likely to be more sensitive to small variations in the prompt.

2.12 Evaluations Based on Static, Human-Written Ground Truth

Another challenge of LLM evaluations is that they often rely on human-written 'ground truth' text. However, we often want to evaluate their performance in domains where such text is scarce or relies on expert knowledge, such as programming or mathematics tasks. As models get more capable and perform better than humans on benchmark tests in some domains, the ability to obtain comparisons to 'human-level' performance diminishes.

Further, benchmark datasets become outdated over time—as models become more capable, older benchmarks become saturated or overfit and no longer provide a useful signal for further improvement [113, 447, 263]. They are typically constructed around a set of tasks that were relevant at the time of creation but may not adapt well to the changing capabilities of LLMs. This means the community must continually adapt to new static benchmarks while de-emphasizing older ones or more dynamic evaluation measures, such as human evaluation of model outputs.

⚠ Reliance on Static, Human-Written Ground Truth

Static benchmarks become less useful over time due to changing capabilities while updating them often relies on human-written ground truth.

To combat these issues, Srivastava et al. [519] regularly admit new tasks to the *Beyond the Imitation Game benchmark* (BIG-Bench), including programmatically evaluated tasks. Further, we highlight two separate streams of work enabling dynamic evaluations without humans in the loop.

Model-generated evaluation tasks As LLM capabilities improve, they can increasingly generate useful benchmark questions or evaluation prompts themselves. Perez et al. [421] shows that LLMs can be used to generate static benchmark datasets for arbitrary axes, using reward models trained on human preferences to filter a generated dataset for quality. Wang et al. [581] find that the order in which candidate examples are presented in the prompt can greatly impact the model-generated evaluation. To mitigate this issue, they propose the usage of a prompting template which encourages the model to generate assessment evidence before assigning a score and averaging scores of multiple assessments with swapped candidate positions.

Model-generated scores Aside from generating evaluation questions, models are increasingly used to directly grade the performance of other models and act as a 'judge' of other models' capabilities [325, 586, 238]. This concept follows the motivation that while it may be challenging for a model

to generate 'correct' answers to prompts in many domains, it can often be easier to evaluate the correctness of an answer or to judge the relative quality between two answers [667, 156]. However, these techniques often produce evaluation results that vary significantly depending on the 'judge' model and suffer from robustness issues that make them a poor substitute for human judgment.

2.13 Indistinguishability between Generated and Human-Written Text

Detecting language generated by LLMs is important for various reasons; some of which include preventing (1) the spread of misinformation (e.g., authoritative-sounding false narratives citing fake studies) [657], (2) plagiarism (e.g., LLMs prompted to rewrite existing content in ways that bypass plagiarism detection tools) [574, 573], (3) impersonation or identify theft (e.g., by mimicking a person's writing style) [486, 602], and (4) automated scams and frauds (e.g., large-scale generation of phishing emails) [603], and (5) accidentally including inferior generated text in future models' training data [439]. However, such detections become less trivial as the fluency of LLMs improves [34].

⚠ Detecting LLM-generated Text

The difficulty in classifying whether a text is LLM-generated or written by a human.

There are primarily two lines of work addressing this problem: (i) *post-hoc detectors*, which aim to classify arbitrary text as being LLM-generated, and (ii) *watermarking* schemes, which modify the text generation procedure to make the detection easier. However, both approaches can be susceptible to *paraphrase attacks*, which we discuss thirdly.

Post-hoc Detectors Gehrmann et al. [168] open-source a tool that visualizes statistically improbable tokens to support humans in detecting generated text artifacts. Bakhtin et al. [34] explore energy-based models to discriminate between real and fake text, including scenarios where the text generator was trained on a completely different dataset than the discriminator. Uchendu et al. [559] examine three authorship attribution problems: (1) were two texts produced by the same method or not; (2) given a text, was it generated by human or machine, (3) which method generated a given text? Mitchell et al. [371] investigate whether a model can detect its own samples by posing a hypothesis: minor rewrites of generated text have lower probability under the model than the original sample, while the same cannot be said about human-written text. Generated passages tend to lie in the negative curvature regions of the model's log probability function. Their method, *DetectGPT*, exploits this hypothesis by approximating that curvature given some samples.

Watermarking Kirchenbauer et al. [268] employ a *watermark*, i.e., a hidden pattern that is imperceptible to humans but algorithmically identifiable, during inference as follows: for each to be generated token, they (1) hash the previous token to seed a random number generator; (2) using that seed, they randomly partition the vocabulary into a "green list" and "red" list, and (3) sample the next token by excluding any token from the red list. In the case of low-entropy tokens, which renders it difficult to introduce changes to the vocabulary, they introduce a "soft" version, which promotes using the green list only for high-entropy tokens (when many plausible choices are available). In follow-up work, the same first authors Kirchenbauer et al. [269] study the robustness of their watermarking scheme *in the wild*, i.e., after it is re-written by humans, non-watermarked LLMs, or mixed into a longer hand-written document. They conclude that watermarks remain detectable given sufficient tokens and argue that this required amount of text is a crucial yet overlooked metric.

Yang et al. [638] study watermarking of black-box API models, where we cannot access the model's inference procedure. Tang et al. [537] provide algorithms for identifying watermarks, noting that watermarked LLMs tend to produce token distributions that differ identifiably from non-watermarked models. Christ et al. [87] introduce *undetectable* watermarks, which can only be detected with the knowledge of a secret key.

To make watermarks robust to text corruptions (we study a common type of such in the next paragraph), Yoo et al. [649] suggest placing them on "invariant features", which are invariant to minor modifications of the text.

Paraphrasing Attacks One way to evade machine-generated text detectors is to re-phrase the text such that the revealing LLM signatures get removed.

⚠ Paraphrasing Attacks

Another LLM can rewrite LLM-generated text to preserve approximately the same meaning but change the words or sentence structure.

Krishna et al. [280] evade several detectors (e.g., dropping DetectGPT's detection accuracy from 70.3% to 4.6%) by training an 11B paraphrase generation model that can paraphrase paragraphs and provides scalar knobs to control the amount of lexical diversity and reordering in the paraphrases. To defend against such attacks, they propose storing model generations in a database, from which the API provider can retrieve semantically similar texts later. Since paraphrasing does not modify the semantics of the text, the authors demonstrate that this retrieval approach is fairly robust to paraphrasing attacks.

Sadasivan et al. [469] claim that the detection of generated text, even with watermarking, is not reliable; neither in practice, by performing paraphrasing attacks; nor in theory, by providing a theoretical impossibility result. They also discuss how an adversary can query watermarked LLMs multiple times to extract its watermarking scheme and spoof the watermark detector by composing human text that is then wrongly classified as model-generated.

2.14 Tasks Not Solvable By Scale

The ongoing advancements of LLM capabilities consistently astonish the research community, for instance, by achieving high performances on the MMLU [197] benchmark much sooner than competitive human forecasters had anticipated [93]. Similarly, within less than a year, OpenAI released GPT-3.5 and GPT-4, where the latter significantly outperformed the former on various tasks [398].

Given this progress, one may question whether there are limits we deem impossible to overcome within the current paradigm of scaling data/model sizes of autoregressive Transformer-based LLMs. We emphasize that such tasks' (permanent) existence is still somewhat speculative. Here, we explore possible patterns behind such tasks instead of discussing specific ones (which we do in Sec. 2.11 and Sec. 3).

⚠ Tasks Not Solvable By Scale

Tasks *seemingly* not solvable by further data/model scaling.

Inverse Scaling (IS) is the phenomenon of task performance worsening as model scale and training loss performance increases. Lin et al. [323] first stumbled upon this property when evaluating models of increasing sizes (e.g., GPT-2, GPT-3) on their benchmark that measures whether an LLM is truthful in generating answers to questions. They conjecture that common training objectives incentive false answers (which they call *imitative falsehoods*) if they have a high likelihood on the training distribution (we discuss dataset issues in Sec. 2.1). McKenzie et al. [359] collect 11 datasets that exhibit IS behavior and identify four potential causes for such: (1) models regurgitating memorized data rather than following in-context instructions, (2) imitation of undesirable patterns in the training data, (3) models learning to perform easier, so-called *"distractor task"* rather than the intended ones, and (4) spurious correlations in the given few-shot examples.

Wei et al. [600] somewhat challenge the existence of inverse scaling by evaluating the tasks proposed by McKenzie et al. [359] on even larger models; up to trained on five times more compute. In this increased compute region, four out of eleven tasks remain inverse scaling; six out of eleven exhibit *"U-shaped scaling"*, where the performance first decreases up to a certain size and then increases again. The authors hypothesize that U-shaped scaling occurs when a task contains a distractor task, which larger models can learn to ignore. Similarly, in the case of quantifier comprehension tasks, Gupta [184] argue that previously observed inverse scaling behavior might have been due to inappropriate testing methodology.

Compositional tasks composed of multiple subproblems are an ideal outlet to investigate whether models go beyond rote memorization of observed facts and deduce novel knowledge [435]. Zhang et al. [661] investigate whether language models can learn deductive reason from data by introducing a class of propositional logic problems. The authors prove that the model has enough capacity to solve the task, yet, it instead learns to rely on statistical features rather than emulating the correct reasoning function. Press et al. [435] measure

how often a model can correctly answer all sub-problems but not generate the overall solution, a ratio they refer to as *compositionality gap*. They find that increasing the model size in the GPT-3 family of models improves solving sub-problems faster than composed problems, suggesting that larger models show no improvement for this gap. Dziri et al. [135] find that systematic problem-solving capabilities do not emerge from maximum likelihood training of Transformer models in general. They base this claim on two hypotheses: (i) Transformers reduce compositional tasks into linearized path matching, a form of shortcut learning [169] that does not generalize robustly; and (ii) errors in the early stages of the task (i.e., when sub-problems follow some order) compound substantially. Asher et al. [26] prove that LLMs cannot learn semantic entailment or consistency as defined in formal semantics [128] due to a lacking understanding of universal quantifiers (e.g., *every, some, many, most, etc.*).

Memorization vs. Generalization An ongoing debate evolves around the question of to what degree LLMs memorize instead of generalize (and what exactly the difference is [35]). Memorization has been shown to (1) hurt (certain) downstream task performances [294], (2) increase with the model size [67, 264, 553, 354], and (3) emerge unpredictably from smaller or partially-trained models [42]. Hence, we wonder whether some tasks do not benefit from further model/dataset size scaling.

One such class of tasks might be *counterfactual tasks* [619], i.e., tasks on which LLMs initially perform well modified such that specific input-output conditions are changed while the general reasoning procedure remains the same. For example, for an arithmetic task, the counterfactual variant would alter the base from 10 to 2. Wu et al. [619] find that LLMs perform poorer the less common the counterfactual conditions are, which they call a *"memorization-like effect"*. An interesting future direction would be to explore whether increasing model size exacerbates performance due to more memorization or actually improves because scaling-law-optimal pre-training recipes would dictate scaling the dataset proportionally (Sec. 2.3), which then may include more of such tasks with uncommon conditions.

2.15 Lacking Experimental Designs

Table 2 shows a (non-exhaustive) overview of selected LLMs within the scope of this review, described in academic papers. Many works do not include controlled ablations, which is especially problematic due to their large design space. We posit that this impedes scientific comprehension and advancement.

Lack of Controlled Ablations We observe that many papers do not run controlled experiments (*ablations*) by varying one factor at a time, likely due to the prohibitive computational cost. For example, Chowdhery et al. [86] conjecture PaLM might outperform GPT-3 and other LLMs on many tasks due to higher training corpus quality, but note they "do not perform the necessary ablation studies to say this conclusively" and instead solely focus on model depth and width. Many papers from Table 2 adopt hyper-parameters from previous works [476] and do not tune them after introducing a change in the training pipeline. Sometimes, important implementation details are not mentioned, e.g., when optimizer states are reset during training [90].

⚠ Uncontrolled Experiments

Papers presenting novel LLMs often lack controlled experiments, likely due to the prohibitive costs of training enough models.

An easy yet expensive fix is to run ablations by varying one factor at a time, e.g., keeping most hyper-parameters fixed except the model size [44] or context lengths [557]. A cheaper potential remedy can be *zero-shot hyper-parameter transfer* from smaller models to larger ones [608, 633]. Yang et al. [633] find that when using the μP network parameterization scheme, one can transfer the effect of changing hyper-parameters such as the learning rate across varying model depths, batch sizes, sequence lengths, and training times, which they verify empirically up to a 6.7B model. However, it has yet to be verified if such transferability still holds for other varying factors; and if so, researchers could afford to conduct more ablation experiments via smaller models.

If additional experiments are prohibitively expensive, another recommendation is to report evaluation results beyond aggregated performance measures. For example, in reinforcement learning, recent work has argued that providing entire perfor-

Table 2: **Overview of selected LLMs**. **Missing details denoted by N/A**. For papers that investigate various model sizes, we only report the largest. For each tokenizer entry with "SP", we could not extract from the respective paper whether BPE or Unigram tokenization was used. For publicly available code repositories and checkpoints, the corresponding ✓ is clickable. Abbreviations: Autoregressive blank filling (ARBF) [132], Byte-pair encoding (BPE), Instruction-following (IF), Masked Language Modeling (MLM), Rotary Next token prediction (NTP), SentencePiece (SP), Span Corruption (SC).

Date	Name	Organization	Language	# Parameters	# Tokens	Architecture	Train. Obj.	Tokenizer	Pos. Embed.	IF	MoE	Code avail.	Ckpt. avail.	Pre-trained
2018.11	GPipe [226]	Google	Multil.	6B	N/A	Enc. & Dec.	NTP	BPE	Learned	✗	✗	✓	✗	✗
2019.09	Megatron-LM [501]	Microsoft	Eng.	8.3B	157B	Dec.-Only	NTP	BPE	Learned	✗	✗	✓	✗	✗
2019.10	T5 [443]	Google	Multil.	11B	1T	Enc. & Dec.	SC	SP	T5	✗	✗	✓	✓	✗
2020.05	GPT-3 [59]	OpenAI	Eng.	175B	300B	Dec.-Only	NTP	BPE	Learned	✗	✗	✗	✗	✗
2020.06	GShard [298]	Google	Multil.	600B	1T	Enc. & Dec.	NTP	SP	N/A	✗	✓	✗	✗	✗
2020.10	mT5 [631]	Google	Multil.	13B	1T	Enc. & Dec.	SC	SP	T5	✗	✗	✓	✓	✗
2021.01	Switch [145]	Google	Multil.	1.5T	N/A	Enc. & Dec.	SC	SP	T5	✗	✓	✓	✓	✗
2021.03	BASE [302]	Meta	Eng.	117B	N/A	Enc. & Dec.	NTP	BPE	Sinus.	✗	✓	✓	✗	✗
2021.04	PanGu-α [659]	Huawei	Multil.	200B	317B	Dec.-Only	NTP	BPE	Learned	✗	✗	✗	✗	✗
2021.05	ByT5 [630]	Google	Multil.	12.9B	1T	Enc. & Dec.	SC	N/A	T5	✗	✗	✓	✓	✗
2021.06	CPM-2 [669]	Tsinghua Uni.	Multil.	198B	N/A	Enc. & Dec.	SC	Custom	Sinus.	✗	✓	✓	✓	✗
2021.06	nmT5 [255]	Google	Multil.	3.7B	100B	Enc. & Dec.	MLM, NTP	SP	T5	✗	✗	✗	✗	✓
2021.07	ERNIE 3.0 [530]	Baidu	Chin.	10B	375B	Enc. & Dec.	Custom	BPE	Rel.	✗	✗	✗	✗	✗
2021.08	Jurassic-1 [319]	AI21	Eng.	178B	300B	Enc. & Dec.	NTP	SP	Learned	✗	✗	✗	✗	✗
2021.08	ExT5 [23]	Google	Eng.	11B	1T	Enc. & Dec.	SC, Custom	SP	T5	✗	✗	✓	✗	✗
2022.01	FLAN-LaMDA [598]	Google	Eng.	137B	245M	Dec.-Only	NTP	BPE	T5	✗	✓	✗	✗	✓
2021.10	M6-10T [322]	Alibaba	Eng.	10T	N/A	Uni. Enc. & Dec.	SC, NTP	SP	N/A	✗	✗	✗	✗	✗
2021.10	Yuan [615]	Inspur AI	Chin.	245B	180B	Dec.-Only	NTP	BPE	N/A	✗	✗	✗	✗	✗
2021.10	T0 [475]	BigScience	Eng.	11B	12B	Enc. & Dec.	SC, NTP	SP	T5	✗	✗	✓	✓	✓
2021.12	Gopher [441]	DeepMind	Eng.	280B	300B	Dec.-Only	NTP	SP	Rel.	✗	✗	✗	✗	✗
2021.12	RETRO [52]	DeepMind	Eng.	7B	419B	Enc. & Dec.	NTP (Ret.)	SP	Rel.	✗	✗	✗	✗	✗
2021.12	GLaM [130]	Google	Multil.	1.2T	600B	Dec.-Only	NTP	SP	Rel.	✗	✓	✗	✗	✗
2021.12	WebGPT [379]	OpenAI	Eng.	175B	N/A	Dec.-Only	NTP	BPE	Learned	✗	✗	✗	✗	✓
2021.12	FairSeq [400]	Meta	Eng.	1.1T	300B	Dec.-Only	NTP	BPE	Sinus.	✗	✓	✓	✓	✗
2021.12	XGLM [324]	Meta	Multil.	7.5B	500B	Dec.-Only	NTP	Unigram	Sinus.	✗	✗	✓	✓	✗
2022.01	LaMDA [551]	Google	Eng.	137B	768B	Dec.-Only	NTP	BPE	T5	✗	✗	✗	✗	✗
2022.01	MT-NLG [515]	Microsoft	Eng.	530B	270B	Dec.-Only	NTP	BPE	Sinus.	✗	✗	✗	✗	✗
2022.02	ST-MoE [687]	Google	Eng.	269B	1.5T	Enc. & Dec.	SC	SP	Sinus.	✗	✓	✓	✗	✗
2022.03	InstructGPT [403]	OpenAI	Eng.	175B	N/A	Dec.-Only	RLHF	BPE	Learned	✓	✗	✗	✗	✓
2022.03	GopherCite [362]	DeepMind	Eng.	280B	N/A	Dec.-Only	RLHF	BPE	Rel.	✓	✗	✗	✗	✓
2022.03	sMLP [653]	Meta	Eng.	9.4B	N/A	Enc. & Dec.	NTP	BPE	Sinus.	✗	✓	✗	✗	✗
2022.03	Chinchilla [206]	DeepMind	Eng.	70B	1.4T	Dec.-Only	NTP	SP	Rel.	✗	✗	✗	✗	✗
2022.04	PaLM [86]	Google	Multil.	540B	780B	Dec.-Only	NTP	SP	RoPE	✗	✓	✗	✗	✗
2022.04	GPT-NeoX [47]	EleutherAI	Eng.	20B	472B	Dec.-Only	NTP	BPE	RoPE	✗	✗	✓	✓	✗
2022.04	Tk-Instruct [589]	AI2	Eng.	11B	1B	Enc. & Dec.	NTP	SP	T5	✓	✗	✓	✓	✗
2022.04	METRO-LM [33]	Microsoft	Eng.	5.4B	2T	Enc.-Only	METRO	SP	T5	✗	✗	✗	✗	✗
2022.04	mGPT [500]	Sber	Multi.	13B	440B	Dec.-Only	NTP	BPE	Learned	✗	✗	✓	✓	✗
2022.05	OPT [666]	Meta	Eng.	175B	300B	Dec.-Only	NTP	BPE	Learned	✗	✗	✓	✓	✗
2022.05	UL2 [545]	Google	Eng.	20B	1T	Enc. & Dec.	MoD	Unigram	T5	✗	✗	✗	✓	✗
2022.05	DeepStruct [578]	UC Berkeley	Eng.	10B	N/A	Enc. & Dec.	Struc.	BPE	Sinus.	✗	✗	✗	✗	✗
2022.07	Minerva [305]	Google	Eng.	540B	26B	Dec.-Only	NTP	SP	RoPE	✗	✗	✗	✗	✗
2022.08	PEER [482]	Meta	Eng.	11B	5B	Enc. & Dec.	NTP	SP	T5	✗	✗	✗	✗	✓
2022.08	AlexaTM [517]	Amazon	Multil.	20B	1T	Enc. & Dec.	MoD, NTP	SP	Sinus.	✗	✗	✗	✓	✓
2022.10	GLM-130B [658]	Tsinghua Uni.	Multil.	130B	400B	Uni. Enc. & Dec.	ARBF	SP	RoPE	✗	✗	✓	✓	✗
2022.10	U-PaLM [547]	Google	Eng.	540B	1.3B	Dec.-Only	MoD	SP	RoPE	✗	✓	✗	✗	✓
2022.10	FLAN-PaLM [93]	Google	Eng.	540B	1.4B	Dec.-Only	NTP	SP	RoPE	✓	✓	✗	✗	✓
2022.11	BLOOM [479]	BigScience	Multil.	176B	366B	Dec.-Only	NTP	BPE	ALiBi	✗	✗	✓	✓	✗
2022.11	Galactica [548]	Meta	Eng.	120B	450B	Dec.-Only	NTP	BPE	Learned	✗	✗	✓	✓	✗
2022.11	Atlas [236]	Meta	Eng.	11B	N/A	Enc. & Dec.	MLM	BPE	T5	✗	✗	✓	✓	✓
2022.11	BLOOMZ [377]	BigScience	Multil.	176B	13B	Dec.-Only	NTP	BPE	ALiBi	✓	✗	✓	✓	✓
2022.11	mT0 [377]	BigScience	Multil.	13B	13B	Enc. & Dec.	NTP	SP	T5	✓	✗	✓	✓	✓
2022.12	OPT-IML [235]	Meta	Eng.	175B	2B	Dec.-Only	NTP	BPE	Sinus.	✓	✗	✓	✓	✓
2022.12	Med-PaLM [511]	Google	Eng.	540B	0B	Dec.-Only	NTP	SP	RoPE	✗	✗	✗	✗	✓
2023.02	LLaMA{-I} [556]	Meta	Eng.	65B	1.4T	Dec.-Only	NTP	BPE	RoPE	✓	✗	✓	✓	✗
2023.03	PanGu-Σ [455]	Huawei	Multil.	1T	329B	Dec.-Only	NTP	BPE	Learned	✗	✓	✗	✗	✓
2023.03	CoLT5 [15]	Google	Eng.	5.3B	1T	Enc. & Dec.	MoD	N/A	T5	✗	✗	✗	✗	✗
2023.03	BloombergGPT [616]	Bloomberg	Eng.	50B	569B	Dec.-Only	NTP	Unigram	ALiBi	✗	✗	✗	✗	✗
2023.04	Cerebras-GPT [121]	Cerebras	Eng.	13B	257B	Dec.-Only	NTP	BPE	RoPE	✗	✗	✗	✓	✗
2023.04	Pythia [44]	EleutherAI	Eng.	12B	300B	Dec.-Only	NTP	BPE	RoPE	✗	✗	✓	✓	✗
2023.04	WizardLM [625]	Microsoft	Eng.	30B	N/A	Dec.-Only	NTP	BPE	RoPE	✓	✗	✓	✓	✓
2023.05	Guanaco [118]	Univ. of Washington	Multil.	65B	82M	Dec.-Only	NTP	BPE	RoPE	✓	✗	✗	✓	✓
2023.04	RWKV [417]	RWKV	Eng.	14B	N/A	Dec.-Only	NTP	BPE	RoPE	✓	✗	✓	✓	✓
2023.06	Orca [378]	Microsoft	Eng.	13B	N/A	Dec.-Only	NTP	BPE	RoPE	✓	✗	✗	✗	✓
2023.07	LLaMA 2 [557]	Meta	Eng.	70B	2T	Dec.-Only	NTP	BPE	RoPE	✓	✗	✓	✓	✓

mance distributions across all runs is less biased and more robust to outliers than point estimates [9].

Curse of Dimensionality In Table 2, we highlight some but not all differences across models, as the table format constrained us. Other common differences include the training datasets or fine-grained architectural details, e.g., the usage of multi-head [563] or multi-query attention [494].

We note that a core characteristic of LLMs is their vast design space, which renders scientific inquiry challenging [231]. For example, by taking into account the (i) data sources and their proportions within the pre-training dataset, (ii) choice and training hyper-parameters of the tokenizer, and (iii) pre-training objective, the combined design space quickly becomes high-dimensional. Undertaking factorial experiments within such expansive design spaces results in a combinatorially-growing number of single training runs, and the lack of sufficient experimental coverage can severely inhibit scientific understanding of what makes an LLM perform well. While this issue is not unique to LLMs, they tend to be larger in the number of parameters—and therefore compute requirements, feedback loop times, and training costs—than models in most other fields.

⚠ Curse of (Design) Dimensionality

Common design spaces of LLM experiments are high-dimensional.

One possible way forward is to encourage the community to use techniques like Bayesian optimization (BO) with dimensionality reduction [594, 374], where we use a non-linear feature mapping to map the input (the hyper-parameter configuration) onto a lower dimensional manifold followed by a BO procedure to optimize the underlying black-box function (the LLM with respect to the hyper-parameters). Another suitable tool to explore the design space efficiently can be treatment effect estimation [284, 385], e.g., where the treatment is a vector describing certain ablations [254].

2.16 Lack of Reproducibility

The reproducibility of empirical results is important to verify scientific claims and rule out errors in experimental protocols leading to such. When researchers try to build upon non-reproducible results, they might waste resources.

Unfortunately, we stumble upon two unique reproducibility issues in LLM research: repeatability of (i) training runs and (ii) generations by close-sourced API-served models. While the term "reproducibility" is often used more broadly and can slightly vary in its meaning [5], in the following, we focus on "repeatability", which we define as the ability to repeat experimental outcomes exactly.

Training Repeatability Typical training protocols of LLMs involve parallelism across multiple compute nodes. The scheduling and communication strategies between nodes can be non-deterministic [387]. This variability can affect the final result, especially in algorithms that are not "order-invariant", such as stochastic gradient descent (SGD). Some sources of randomness are (i) lock-free parallelism schemes [387], (ii) floating point precision, e.g., when summing gradients across devices, the order in which these sums are computed can affect the final result [171], (iii) non-deterministic, performance-optimized operations, which are much faster and therefore desirable [3].

Further, Carlini et al. [64] point out that some pre-training datasets consist of an index of web content that individual users must crawl themselves, rather than using static, standalone dumps. This is due to monetary, privacy, and legal restrictions. As a result, reproducibility can be easily compromised if any of the sources in the index have changed between the time the dataset curator collected them and the time the end-user downloads them.

⚠ Irrepeatable Training Runs

Parallelism strategies designed to distribute the training process across many accelerators are typically non-deterministic, rendering LLM training irreproducible.

Inference Repeatability Another peculiarity of commercial LLMs is that they are typically served via stochastic API in a black-box setting, which comes with the following challenges: (i) the provider retains complete authority over the model and can introduce unpublicized changes, including retraining the model, modifying its parameters, or completely replacing it; (ii) even if model updates are communicated, there is still uncertainty about whether access to specific model versions will be maintained once they are deemed outdated, (iii) even with a decoding temperature

set to zero, API models often produce stochastic outputs [392, 464, 456].

Chen et al. [76] provide preliminary evidence confirming dramatic changes in API-served models. They find that GPT-3.5 and GPT-4 performances on four diverse tasks vary vastly within three months (March to June 2023). For example, GPT-4's accuracy in identifying prime numbers was 97.6%, but in June, its accuracy dropped to 2.4%; while for GPT-3.5, the trend is reversed and it got much better over time.

⚠ Irreproducible API Inference

API-served models are often irreproducible.

An easy fix is to rely exclusively on open-source LLMs [2].

3 Applications

In this section, we aim to provide practitioners with a broad overview of the areas in which LLMs are currently being applied and highlight some common application architectures across domains.

Analogous to the Challenges section, we highlight the key constraints in each application area as follows.

⚠ Constraint

This box highlights a constraint.

3.1 Chatbots

General-purpose chatbots (dialogue agents) combine the tasks of information retrieval, multi-turn interaction, and text generation (including code).

Thoppilan et al. [551] introduced the LaMDA family of chatbot LLMs with up to 137B parameters, focusing on safety (via supervised fine-tuning on human annotations) and factual grounding (via access to external knowledge sources). Notably, smaller LaMDA models (2B parameters) with fine-tuning are shown to perform similarly on dialogue quality and safety/grounding scores to the larger LaMDA models (137B parameters) without fine-tuning. LaMDA models were released as part of the Bard chatbot service [429]. However, the latest version of Bard now uses the PaLM 2 LLM [20, 216].

Glaese et al. [170] propose Sparrow, a chatbot based on a 70B parameter Chinchilla LLM, and use RLHF (Sec. 2.9) targeting 23 rules to fine-tune the model to be more *helpful*, *correct*, and *harmless*. Sparrow also incorporates external knowledge using a retrieval model to provide evidence from a Google Search query. The RLHF approach outperforms the only dialogue-prompted and supervised fine-tuned approaches regarding output preference and rule violation rate.

Similarly, OpenAI [396] train the ChatGPT chatbot using supervised fine-tuning and RLHF (Sec. 2.9) to specialize a GPT-3.5 LLM for dialogue. GPT-4 [398] is the underlying model for the ChatGPT Plus chatbot, but training and architecture details have not been released.

Shuster et al. [508] introduce BlenderBot-3, a 175B parameter chatbot based on the OPT-175 LLM using supervised fine-tuning. BlenderBot-3 incorporates external knowledge through modules that conduct internet searches and retrieve text-based long-term memories generated from previous outputs to help performance over long interactions.

⚠ Maintaining Coherence

Multi-turn interactions make Chatbots easily "forget" earlier parts of the conversation or repeat themselves [53, 451].

Köpf et al. [274] release the OpenAssistant Conversations dataset of human-annotated interactions and use this to instruction fine-tune Pythia and LLaMA models (up to 30B parameters) for chatbot applications. To help align the final models, the dataset is generated with guidelines to make the responses *polite*, *helpful*, *concise*, *friendly*, and *safety-aware*. The LLaMA 30B version is currently used within the HuggingChat chatbot application [229].

A key challenge of fine-tuning chatbots is creating a broad training dataset of high-quality conversations. To address this problem Chen et al. [78] demonstrate using existing LLMs (OPT 30B) to generate high-quality synthetic conversation datasets based on a small number of expert-written examples. Human crowd workers assessed the generated conversations to be comparable to existing human-generated datasets on the metrics: *interesting*, *coherent*, *natural*, and *consistent*. Chen et al. [78] show the synthetic dataset can be used to fine-tune a chatbot (BlenderBot 400M) and achieve performance only slightly below fine-tuning with human-generated datasets.

Chatbots' intended generality also makes eval-

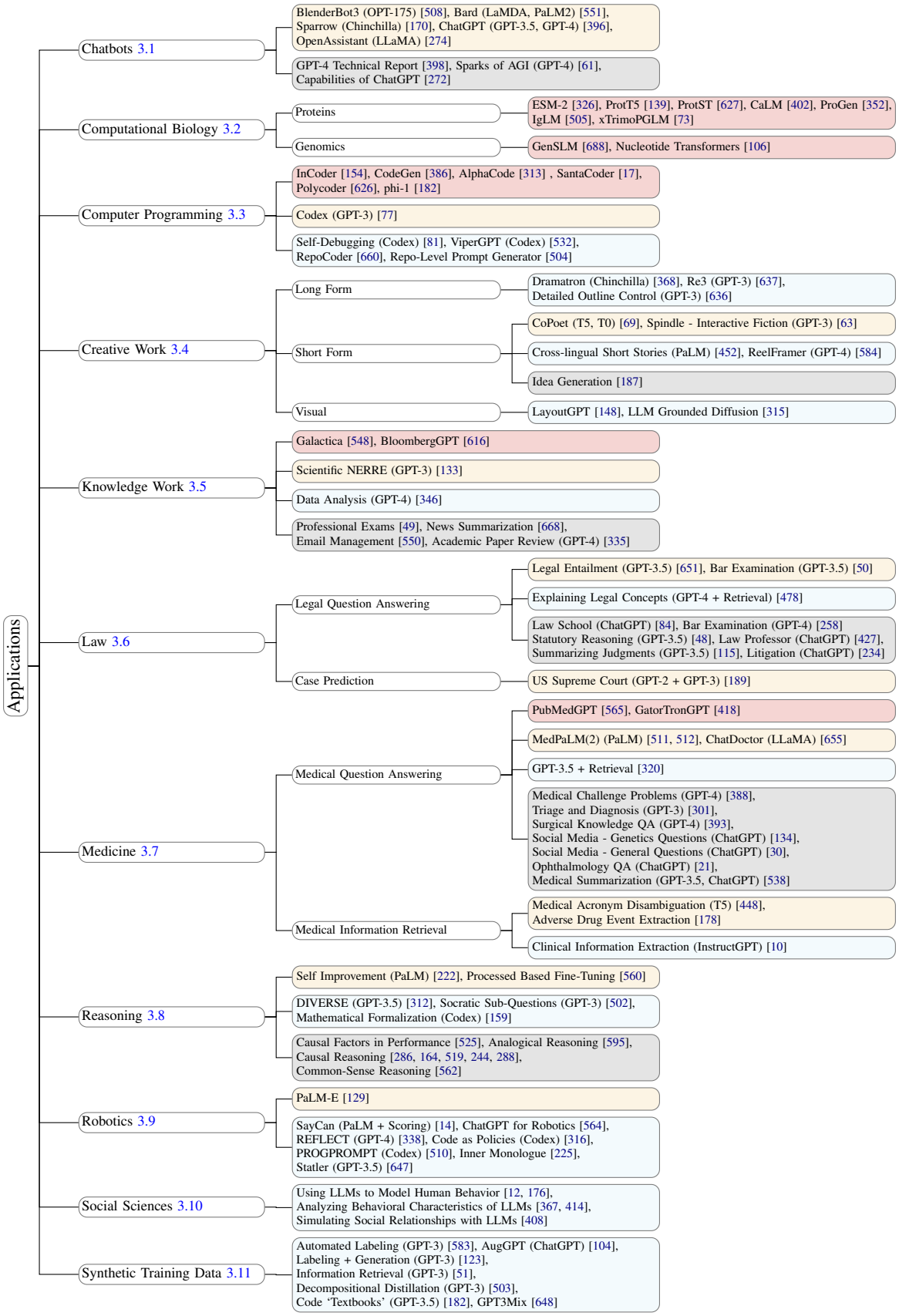


Figure 12: Overview of LLM Applications. Color = Level of Model Adaption (Pre-Trained, Fine-Tuned, Prompting Strategy, Evaluation).

uating their capabilities' full range difficult. Kocoń et al. [272] evaluate ChatGPT (GPT-3.5) on 25 tasks with 38k prompts covering a diverse set of capabilities, including but not limited to question answering, emotion recognition, offensive language detection, spam detection, inference, and sentiment analysis. While ChatGPT is shown to have strong performance across the 25 tasks, it usually underperforms the SOTA in that domain. More recently, Bubeck et al. [61] and OpenAI [398] investigate the capabilities of GPT-4 (base model of ChatGPT Plus) across a wide range of tasks, including interactions with humans and tools. Using these evaluations Bubeck et al. [61] conclude that GPT-4 is 'strikingly close to human-level performance' across tasks.

Finally, the challenge of inference latency (Sec. 2.5) is also potentially going to become an important constraint [634] for chatbot applications as LLMs scale. There is a trade-off between the need for responsive live user interaction in a conversational format and utilizing larger LLMs [397].

⚠ High Inference Latency

High inference latency (Sec. 2.5) hinders the user experience [397], especially in multi-turn interaction with chatbots.

3.2 Computational Biology

In computational biology, we are interested in non-text data representing similar sequence modeling and prediction challenges.

3.2.1 Protein Embeddings

One popular application of LLM-like models in biology is to generate protein embeddings from amino-acid or genomic sequence inputs. These embeddings can then be used as inputs for structure prediction, novel sequence generation, and protein classification tasks. Protein language models perform strongly on many academic datasets, but their applicability to downstream tasks such as drug design is often unclear [110].

⚠ Transfer to Downstream Applications

The ultimate objective of protein language models is to deploy them in real-world projects such as drug design. Evaluations often target smaller and/or specialized datasets, not considering how the models could contribute to protein design in vitro or in vivo.

Elnaggar et al. [139] train a range of LLM architectures to extract embeddings from protein amino acid sequences. These embeddings are then used as inputs on supervised per-amino acid and per-protein prediction tasks. The best-performing LLM architecture (ProtT5) achieved SOTA results on per-amino acid protein secondary structure prediction without using evolutionary information. Similarly, Wu et al. [613] predict antibody backbone and side-chain conformations.

Lin et al. [326] take a similar approach to training a protein LLM, the Evolutionary Scale Model Transformer-2 (ESM-2), on protein amino acid sequences from the UniRef database using a masked language modeling approach. They show significant performance increases as the model is scaled from 8 million to 15B parameters, with the largest models outperforming the ProtT5 on protein structure prediction benchmarks (CASP14, CAMEO) [267, 457]. They also introduce ESMFold, which uses the ESM-2 embedding model for end-to-end atomic resolution prediction from a single sequence. While ESMFold underperforms the SOTA AlphaFold2 [248] on the CAMEO and CASP14 benchmarks, the authors note that by relying only on embeddings ESMFold has an order of magnitude faster inference time than AlphaFold2, using just the protein sequence of interest rather than structural templates and multiple sequence alignments (MSAs). Jeliazkov et al. [240] find that protein sequences designed by an inverted AlphaFold2 model are unlikely to be expressed, but sequences generated using an inverted protein LLM such as ESMFold were more likely to be expressed.

Researchers have also adopted the ESM-1 and ESM-2 models to generate protein embeddings for enzyme-substrate chemical structural class prediction [245], training 3D geometric graph neural networks for proteins [611], identifying disease-causing mutations [337], designing novel proteins [566], and guided evolution of antibodies for affinity maturation [202].

Chen et al. [73] propose training a new model xTrimoPGLM (100B parameters) simultaneously for protein embedding and generation tasks using MLM and generative objectives. The xTrimoPGLM-100B model (with fine-tuning where relevant) outperforms existing approaches on 13 out of 15 evaluated tasks.

Protein embedding models with alternative inputs have also been proposed. Outeiral and Deane [402] train an 86 million parameter protein LLM CaLM (Codon adaptation Language Model) using sequences of codons (nucleotide triads) as input instead of amino acids due to codons containing potentially richer information. Madani et al. [352] train a 1.2B parameter protein embedding model ProGen on 280 million protein amino acid sequences with additional *control tags* specifying protein properties. ProGen is then fine-tuned using data from specific protein families and applied to generate functional full-length amino acid sequences. Similarly, Xu et al. [627] propose training a protein language model, the ProtST, on protein sequences and additional text descriptions of their key properties for protein classification and retrieval tasks.

Finally, for antibodies specifically, Shuai et al. [505] propose an Immunoglobulin Language Model (IgLM) using the GPT-2 architecture (with 13 million parameters) for the generation of immunoglobulin sequences, using a masked language modeling approach. Similar to Xu et al. [627], the IgLM model also takes additional conditioning tags corresponding to chain type and species as input. The authors show the IgLM model can then be used for the controllable generation of infilled and full-length antibody sequences.

3.2.2 Genomic Analysis

LLMs in the field of genomic analysis enable a better understanding of the effects of mutations in humans and predict genomic features directly from DNA sequences. While genomic language models are a promising research direction, current models cannot process many genomic sequences as their sequence lengths commonly exceed multiple billions of nucleotides [390].

⚠ Limited Context Window

The largest genomes have vastly longer DNA sequences [390] than existing genomic LLMs' context windows can handle, constraining the types of genomes that can be successfully modeled using these approaches.

Zvyagin et al. [688] introduce a range of hierarchical LLMs (up to 25B parameters) with long input sequences (2048 - 10,240 tokens), referred to as Genome-scale Language Models (GenSLMs). The GenSLM models are pre-trained on Prokaryotic gene sequences from the BV-BRC dataset using codon tokenization [402] and then fine-tuned on SARS-CoV-2 genome sequences for the task of identifying potential new variants and generative modeling. However, the authors note that it remains unclear whether the GenSLM architecture generates richer representations than the protein LLM approaches.

Dalla-Torre et al. [106] train Nucleotide Transformers with 500 million to 2.5B parameters on nucleotide sequences from human and other species genomes, using a masked language modeling approach. The Nucleotide Transformers were evaluated on 18 genomic prediction tasks with fine-tuned larger models achieving the best results.

Nguyen et al. [383] propose HyenaDNA, a genomic language model based on the Hyena architecture [430], enabling modeling of genomic sequences of up to 1 million tokens. HyenaDNA outperforms Transformer-based models with multiple orders of magnitude more parameters while incorporating the in-context learning capabilities of LLMs into the genomics domain.

3.3 Computer Programming

One of LLMs' most advanced and broadly adopted applications is generating and completing computer programs in various programming languages. This section deals with programming-specific LLMs where the model is fine-tuned or pre-trained exclusively for programming applications, but it is important to note the increasing use of general chatbots partially trained on code datasets (such as ChatGPT) for programming tasks.

3.3.1 Code Generation

Code generation refers to using an LLM to output new code for a given specification or problem pro-

vided as a prompt. Several computer programming-specific LLMs and approaches have been proposed.

For Python code generation, Chen et al. [77] introduce Codex, a fine-tuned GPT-3 LLM (up to 12B parameters) specialized to generate standalone Python functions from doc strings. Fine-tuning was conducted using a raw dataset of 159 GB of Python source code from GitHub and a filtered dataset of correctly implemented standalone Python functions. Codex models outperformed similarly sized GPT-3 and GPT-J models on the HumanEval evaluation set, with the Codex model trained on the filtered dataset (Codex-S) achieving the best results. Importantly, Chen et al. [77] note that there was no observed improvement from using a pre-trained GPT-3 model as a base other than faster convergence.

Chen et al. [81] seek to improve the performance of Codex through a *self-debugging* prompting approach. Three forms of *self-debugging* are investigated. *Simple* feedback prompts the model to decide whether the generated code solution is correct. *Unit-test* feedback prompts the model with the output of unit tests provided in the problem description. *Code explanation* feedback prompts the model to explain the solution in detail and use the explanation to correct the solution. In each case, this process is repeated iteratively until the model provides a solution it states is correct or a maximum number of attempts has been made. Codex using the *self-debugging* prompting framework with code explanation (and unit-testing if applicable) outperforms the base Codex model on C++-to-Python translation, text-to-SQL generation, and text-to-Python generation.

Gunasekar et al. [182] train a smaller model Phi-1 (1.3B parameters) to generate Python functions from doc strings. Training phi-1 using a combination of filtered existing datasets and new synthetic *textbook* and *exercise* datasets results in a model that can achieve near current SOTA results on HumanEval while having over an order of magnitude fewer parameters and tokens than previous works.

Another area of interest has been the development of multilingual programming LLMs. Xu et al. [626] evaluate a range of code generation LLMs and train a new multilingual LLM Polycoder (2.7B parameters) using source code from 12 languages. However, for Python specifically, Codex outperforms Polycoder and other existing models (GPT-J, GPT-Neo, and CodeParrot) on HumanEval.

Nijkamp et al. [386] train the CodeGen family of LLMs (up to 16B parameters) using a combination of three datasets: natural language, multilingual programming source code (C, C++, Go, Java, JavaScript, and Python), and a monolingual Python dataset. The largest CodeGen model using the monolingual training set was shown to outperform the Codex-12B model. Nijkamp et al. [386] also test CodeGen on multi-step program synthesis, where a program is broken down into multi-step natural language prompts, which the model then implements individually (creating the new Multi-Turn Programming Benchmark (MTPB)).

Finally, Li et al. [313] focus on the task of solving competitive programming questions (Codeforces, Description2Code, and CodeNet). The AlphaCode LLM (up to 41B parameters) is first pre-trained on a multilingual dataset (C++, C#, Go, Java, JavaScript, Lua, PHP, Python, Ruby, Rust, Scala, and TypeScript) of 715 GB of source code from GitHub. It is then fine-tuned using a new curated dataset of competitive programming problems called CodeContests. To achieve high performance, Li et al. [313] use large-scale sampling (up to millions of samples), filtering, and clustering of candidate solutions generated by AlphaCode to select the final submissions.

However, whilst these existing code-generation LLMs have achieved impressive results, a critical current constraint in applying LLMs to code generation is the inability to fit the full code base and dependencies within the context window. To deal with this constraint, a few frameworks have been proposed to retrieve relevant information or abstract the relevant information into an API definition.

⚠ Long-Range Dependencies [660, 504]

Long-range dependencies across a code repository usually cannot be regarded because of limited context lengths (Sec. 2.6).

Zhang et al. [660] introduce RepoCoder, a retrieval-based framework for repository-level code completion that allows an LLM to consider the broader context of the repository. A multi-step *retrieval-augmented generation* approach is taken, where the initial code generated is then used to retrieve further, potentially more relevant, repository code snippets to refine the final output. This approach can be considered a retrieval-based method

for relieving the long-range dependency constraint.

Similarly, Shrivastava et al. [504] propose the Repo-Level Prompt Generator (RLPG) framework to dynamically retrieve relevant repository context and construct the correct prompt for a given completion task. To do this, many *prompt proposals* are generated from different *prompt sources* (e.g., parent class) and *prompt contexts* (e.g., method names). The best prompt is then selected by a *prompt proposal classifier* and combined with the default context to generate the final output.

Finally, Surís et al. [532] create the ViperGPT framework, which utilizes the Codex LLM to generate programs that answer text-based visual queries. The Codex model is prompted with the query text and an API specification to do this. The human-generated API specification provides functions designed to deal with low-level visual tasks (e.g., find(*object*)) that the LLM can then use to generate solution code. This approach significantly reduces the tokens needed to provide repository/-code context by only providing the API definition. This *API definition* approach, illustrated in 13 has been used in robotics by Vemprala et al. [564], and by Wang et al. [579] as part of a Minecraft agent. Previously, Gupta and Kembhavi [185] used a predefined function approach within VISPROG, which uses GPT-3, external python *modules*, and few-shot prompting with example programs to solve visual tasks.

3.3.2 Code Infilling and Generation

Code infilling refers to modifying or completing existing code snippets based on the code context and instructions provided as a prompt.

Fried et al. [154] train the InCoder LLM (up to 6.7B parameters) to both generate Python code and infill existing code using a masked language modeling approach. Incoder is trained using 159 GB of text split roughly equally between Python source code, StackOverflow content, and source code in other languages. On the HumanEval generation benchmark, InCoder underperforms the best-performing Codex and CodeGen models. However, unlike the other models, InCoder can perform single and multi-line infilling of existing code.

Similarly, Allal et al. [17] train a set of smaller SantaCoder models (1.1B parameters) for code generation and code infilling using 268 GB of Python, JavaScript, and Java source code. SantaCoder is primarily evaluated on the MultiPL-E benchmark (an extension of HumanEval and MBPP [28] benchmarks), with it shown to outperform InCoder on both HumanEval generation and infilling (passing over 100 attempts).

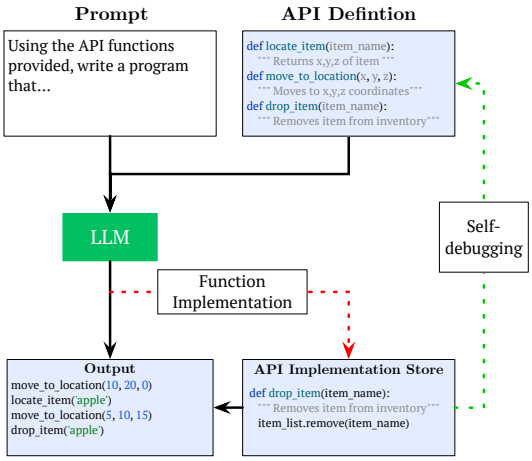


Figure 13: **API Definition Framework**. Illustration of providing a general API definition in the prompt [532, 579, 564] to enable the consistent use of either external code or tools to solve the specific task whilst minimizing the required context window. Extensions to this approach have included asking the LLM to implement the functions within the API definition (red) and to prompt the LLM to self-debug any API code that does not execute (green).

Code infilling is particularly relevant for applications involving modifying, reviewing, or debugging existing code. Maniatis and Tarlow [357] explore the data from the intermediary steps in the development process to help automatically resolve reviewer comments [155]. The Dynamic Integrated Developer ACTivity (DIDACT) methodology formalizes tasks in the software development process (e.g., repairing builds, predicting reviewer comments, etc.) into *state*, *intent*, and *action* components, and trains the model to predict code modifications. This approach aims to train the model to understand the *process* of software development rather than only the end product.

3.4 Creative Work

For creative tasks, LLMs have primarily been applied to story and script generation.

For long-form story generation, Mirowski et al. [368] propose using a 70B Chinchilla-optimal [206] LLM Dramatron with prompting, prompt chaining, and hierarchical generation to create complete scripts and screenplays without the requirement for a human-in-the-loop (although co-writing is facilitated). The ability of Dramatron to help create a script was evaluated qualitatively

through co-writing and follow-up interviews with 15 industry experts.

Similarly, Yang et al. [637] propose using GPT-3 with a Recursive Reprompting and Revision framework (Re3) to generate stories over 2,000 words long. The Re3 approach uses zero-shot prompting with GPT-3 to generate a plan (settings, characters, outline, etc.). It then recursively prompts GPT-3 to generate story continuations using a specified dynamic prompting procedure. Possible story continuations are then ranked for coherence and relevance using separate fine-tuned Longformer models as part of a *Rewrite* module. Finally, local edits to the selected continuations are made by detecting factual inconsistencies using the combination of a GPT-3 model [403] and a BART model [303] as part of an *Edit* module. This process can then be iterated for fully automated story generation.

Finally, Yang et al. [636] introduce the Detailed Outline Control (DOC) framework to maintain plot coherence over thousands of words using GPT-3. While DOC uses the same high-level *planning-drafting-revision* approach as Re3, it implements this through the use of a *detailed outliner* and *detailed controller*. The *detailed outliner* first breaks down the high-level outline into subsections using a breadth-first approach, with candidate generations for the subsections created, filtered, and ranked. The bodies of the detailed outline subsections are then generated iteratively using a structured prompting approach. During the generation, an OPT-based FUDGE [635] *detailed controller* is used to help maintain relevance.

In each case, to apply LLMs to long-form story generation, the task is broken down into a series of short-form sub-tasks (14). The current capabilities of LLMs primarily drive this approach, but also the desire to have a human-in-the-loop for some co-writing use cases [368].

⚠ Limited Context Window [368, 637]

The inability of current LLMs to keep the entire generated work within the context window currently constrains their long-form applications and generates the need for modular prompting (14).

For short form generation, Chakrabarty et al. [69] propose CoPoet (fine-tuned T5 and T0 models) for collaborative poetry generation, Razumovskaia et al. [452] use PaLM and prompting with plans for cross-lingual short story generation, Wang et al. [584] use GPT-4 as part of the ReelFramer tool to help co-create news reels for social media, Ippolito et al. [232] use LaMDA as part of the Wordcraft creative writing assistant, and Calderwood et al. [63] apply a fine-tuned GPT-3 model as part of their Spindle tool for helping generate choice-based interactive fiction.

For more general creative tasks, Haase and Hanel [187] assess a range of LLMs (including ChatGPT) on their capacity for idea generation (*everyday creativity*) using the Alternative Uses Test (generating alternative uses for given items). On this task, LLMs were found to perform comparably to 100 human participants.

Finally, for visual creative tasks, LLMs have also been used to increase the level of control users have when using image generation models. Feng et al. [148] propose the LayoutGPT method where an LLM (GPT-3.5, GPT-4 or Codex) is used to generate a CSS Structure layout the image should follow based on a text-based user prompt. This layout can be visualized and used as input to guide an image generation model. This approach performs strongly on text-to-image generation and indoor scene synthesis. A similar concept is implemented by Lian et al. [315], where an LLM (GPT-3.5) is used to generate natural language layouts (bounding boxes and descriptions) to guide a diffusion model. Using an LLM as part of a *modality conversion* framework 16 has also been used in robotics [338, 225] and knowledge work [329].

3.5 Knowledge Work

With researchers increasingly demonstrating LLMs' ability to perform well on domain-specific knowledge tasks such as within Law [258] or Medicine [512], interest has grown in LLMs' capacity for wider *knowledge work*. These applications are likely to be found across the labor market with Eloundou et al. [140] estimating that 80% of the US workforce is in roles where at least 10% of tasks could be affected by LLMs.

In the professional services field, Bommarito et al. [49] evaluate GPT-3.5 and previous GPT versions on actual and synthetic questions from the Uniform CPA Examination Regulation section and AICPA Blueprints for legal, financial, accounting, technology, and ethical tasks. Using only zero-shot prompting, the best performing model (latest GPT-3.5) struggles with quantitative reasoning, achiev-

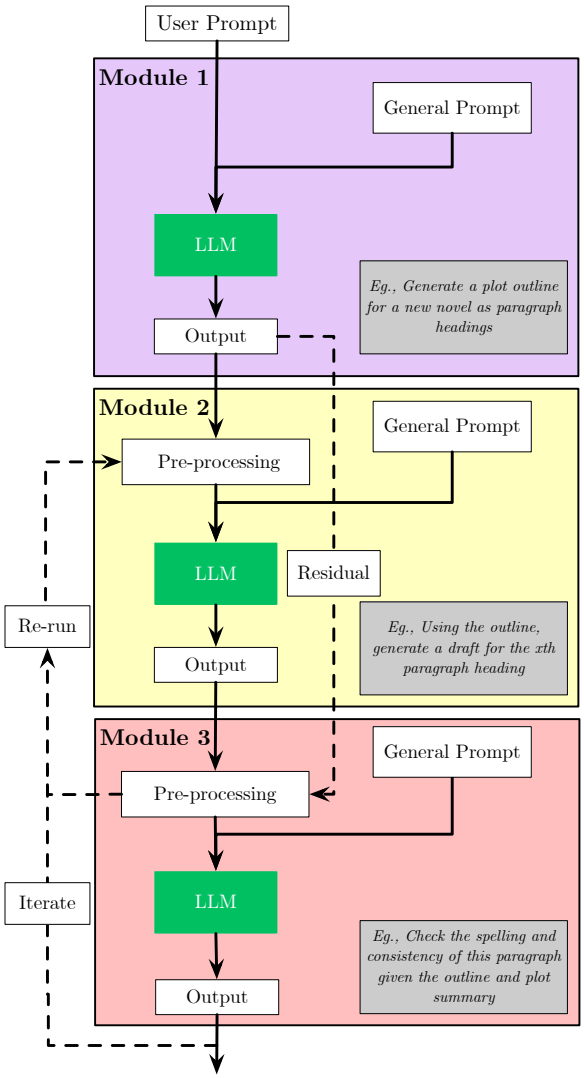


Figure 14: **Modular Prompting**. Illustration of using a series of separate prompts [368, 637, 368, 579, 584] and processing steps to enable an LLM to perform tasks that would either not fit in a single context window or could not easily be specified in a single prompting step.

ing results similar to random guessing on multiple-choice questions. However, on qualitative sections, GPT-3.5 achieved 50-70% accuracy, significantly ahead of random guessing and approaching human-level scores.

⚠ Numerical Reasoning [436, 49]

LLMs have generally seen worse performance on quantitative tasks, potentially constraining their applications in knowledge work areas such as financial services or accounting.

Wu et al. [616] train BloombergGPT (50B parameters) for various financial knowledge work, including sentiment analysis, classification, NER/NED, and financial question answering. BloombergGPT is shown to outperform the OPT (66B parameters), GPT-NeoX, and BLOOM (176B parameters) LLMs on these financial domain-specific tasks and performs competitively on broader benchmarks.

Thiergart et al. [550] considers the applicability of GPT-3 to the task of email management, including classification, information extraction (NER), and generating response text. Whilst it is noted that GPT-3 has the capacity for all three tasks, the author highlights current issues around reliability, lack of access to internal data, and the need for a human in the loop.

Liu et al. [329] propose enabling LLMs to understand charts and plots by first using a vision plot-to-text translation model (DePlot) to decompose the chart into a linearized data table. Once the chart or plot has been converted into a text-based data table, it is combined with the prompt and provided to a Flan-PaLM, Codex, or GPT-3.5 LLM. A similar *modality conversion* 16 approach has also been used in robotics [338, 225] for sensor data.

Zhang et al. [668] evaluate a range of LLMs (GPT-3, InstructGPT, OPT, GLM, Cohere, and Anthropic) on the task of news summarization. On the DM/CNN and XSUM benchmarks, instruction fine-tuned models (InstructGPT) perform the best across summarization faithfulness, relevance, and coherence. To evaluate against human capability Zhang et al. [668] collect reference summarizations for 100 articles from 6 freelance writers. Zero-shot InstructGPT-3 performs comparably to the freelance writers across the three metrics.

Cheng et al. [82] investigate GPT-4's capacity to perform data analysis and compare it to human analysts. GPT-4 is combined with a *modular prompting* framework 14 with three steps, code generation (SQL and Python), code execution ("collect data and output figures", etc.), and analysis generation ("generate five bullet points about the analysis"). While GPT-4 performs well, it currently underperforms experienced human data analysts on tasks from NvBench [346].

For scientific knowledge work, Taylor et al. [548] train the Galactica LLM specifically on scientific text for tasks such as scientific knowledge recall, reasoning, citation prediction, and scientific Q&A. In addition to a domain-specific training corpus, Galactica is specialized in the scientific do-

main through the use of specialized tokens, working memory, and *prompt-pre-training*.

Dunn et al. [133] propose fine-tuning GPT-3 for scientific combined named entity recognition and relation extraction (LLM-NERRE). First, 100 to 1,000 manually annotated prompt-completion pairs are created by humans. These examples are then used to fine-tune a GPT-3 model for the specific NERRE task.

Finally, Liu and Shah [335] evaluate GPT-4's ability to review academic papers, specifically: identifying errors, verifying author checklists, and selecting the *better* abstract. GPT-4 shows some capacity to detect errors, with 7 out of 13 errors detected, and verify author checklists, with 87% accuracy. However, GPT-4 is shown to have limited capacity for distinguishing the *better* paper abstract.

3.6 Law

Applications of LLMs within the legal domain share many similarities with medicine, including legal question answering [651, 258] and legal information extraction [71]. However, other domain-specific applications have been proposed, such as case outcome prediction [189], legal research [234], and legal text generation [423].

3.6.1 Legal Question Answering and Comprehension

Key tasks of the legal field are finding related precedents, answering legal questions, and comparing existing documents or statutes.

Using a general-purpose LLM with prompting approach, Yu et al. [651] use GPT-3.5 with zero-shot, few-shot, and CoT prompting to achieve SOTA performance on the legal entailment task (identifying the relevant statutes and determining if a given premise is correct) in the Competition on Legal Information Extraction/Entailment (COLIEE) dataset [437]. They also investigate a GPT-3.5 version fine-tuned using the COLIEE training set with and without explanations but find the zero- and few-shot legal prompting approaches perform best. Similarly, Rosa et al. [460] use a general monoT5 model with zero-shot prompting on the COLIEE entailment task.

On the US legal Uniform Bar Examination (UBE), Bommarito II and Katz [50] show that GPT-3.5 with zero-shot prompting can achieve 50% on the multiple choice Multistate Bar Examination component, but note that fine-tuning the model on relevant examples does not appear to improve performance. More recently, Katz et al. [258] show that GPT-4 with zero-shot prompting exhibits SOTA performance on the full UBE, including the multiple choice, essay, and performance test components, and achieves passing scores.

Blair-Stanek et al. [48] assess GPT-3.5's ability to reason about legal facts and statutes using the StAtutory Reasoning Assessment (SARA) dataset [208]. GPT-3.5 is shown to have SOTA performance but with significant variation depending on the type of prompting used (zero-shot, few-shot, and CoT). GPT-3.5 was also shown to perform relatively poorly on synthetic statutory reasoning tasks.

Choi et al. [84] evaluate ChatGPT (GPT-3.5) on 95 multiple-choice and 12 essay questions from the final exams at the University of Minnesota law school. ChatGPT was found to perform at the level of a C+ student, near the bottom of the class, but with passing scores.

⚠ Out of Date Information

Due to regularly updated laws and new precedents, the training/retrieval data become outdated frequently [195].

Finally, many more specific legal question-answering applications have been proposed, including: explaining legal concepts (GPT-4 + retrieval) [478], summarizing legal judgments (GPT-3.5) [115], litigation research and drafting [234], and helping full-fill the tasks of a law professor (ChatGPT) [427].

3.6.2 Case Prediction and Legal Text Generation

Case prediction and legal text generation involve predicting or completing legal opinions. Whilst there is currently sparse usage of LLMs in the literature, smaller language models have been applied, suggesting potential future LLM applications in this area.

Hamilton [189] use nine separate GPT-2 models trained on individual supreme court justice's authored opinions to predict how each justice will vote on a given case. They use a handcrafted prompt, including a summary of the topic generated by GPT-3. However, they find this approach to case prediction does not match the SOTA.

Previously, Chalkidis et al. [70] trained a range of attention-based models (including BERT) to pre-

dict case outcomes from the European Court of Human Rights (ECHR). The attention-based models outperformed an SVM with a bag of words approach for binary violation classification, multi-label violation classification, and case importance prediction.

Finally, Peric et al. [423] use a dataset of 50,000 judicial opinions from U.S. Circuit Courts to train a Transformer-XL model and fine-tune a GPT-2 model. The models were then evaluated for their ability to complete a judicial opinion, with a start given as a prompt. In qualitative evaluations, human participants struggled distinguishing between machine-generated and genuine text.

3.7 Medicine

Many applications of LLMs have been proposed in the medical domain, including medical question answering [511, 512, 320, 655, 388], clinical information extraction [10, 448], indexing [650], triage [491, 301], and management of health records [276].

3.7.1 Medical Question Answering and Comprehension

Medical question answering and comprehension consists of generating multiple-choice and free-text responses to medical questions.

Singhal et al. [511] proposed using few-shot, CoT, and self-consistency prompting to specialize the general-purpose PaLM LLM to medical question answering and comprehension. They demonstrate a Flan-PaLM model [93] using a combination of the three prompting strategies to achieve the previous SOTA results on the MedQA, MedMCQA, PubMedQA, and MMLU medical datasets. To further align the model to the medical domain, they proposed Med-PaLM, which utilizes instruction prompt-tuning based on 40 examples from a panel of clinicians and task-specific human-engineered prompts.

Singhal et al. [512] then extend the Med-PaLM approach with Med-PaLM 2 using the newer PaLM 2 LLM as its base model. Singhal et al. [512] conduct further instruction-fine tuning and use a new ensemble refinement (ER) prompting strategy (where stochastically sampled outputs are first generated and provided within the final prompt). This allows Med-PaLM 2 to achieve the current SOTA on the MultiMedQA benchmark.

Liévin et al. [320] adopt a similar approach using zero-shot, few-shot, and CoT prompting to adapt the GPT-3.5 LLM to medical question answering (USMLE and MedMCQA) and comprehension (PubMedQA) tasks. In addition, Liévin et al. [320] propose using retrieval augmentation where relevant text from Wikipedia is retrieved and included in the prompt. More recently, Nori et al. [388] evaluated GPT-4 on USMLE and MultiMedQA datasets using zero and few shot prompting. GPT-4 is found to outperform GPT-3.5 across benchmarks significantly. However, several issues relating to using GPT-4 for real-world clinical applications are raised, including the *risks of erroneous generations* and the *risks of bias*. Tang et al. [538] raise similar issues and find that GPT-3.5 and ChatGPT have issues with factual accuracy and representing the level of certainty during medical summarization.

⚠ Hallucination and Bias [538, 388, 511]

The safety-critical nature of the medical domain means the possibility of hallucinations significantly limits the current use cases. Further work is also needed to reduce the risk of LLMs perpetuating existing bias in clinical datasets.

Yunxiang et al. [655] fine-tune a LLaMA LLM ChatDoctor (7B parameters) specifically for the task of medical question answering. To specialize the LLaMA model, it is first instruction fine-tuned using the Alpaca dataset [540] and then fine-tuned to the medical domain using a dataset of 100k patient conversations. Similarly to Liévin et al. [320], ChatDoctor is augmented with two external knowledge sources (a disease database and Wikipedia) to improve the factual grounding of the model.

Instead of using general models with specialized prompting or fine-tuning, Venigalla et al. [565] train a new model PubMedGPT specifically for medical question answering and text generation tasks. PubMedGPT is trained using a combination of PubMed abstracts and full documents from the Pile [165]. Peng et al. [418] also train a new LLM GatorTronGPT (up to 20B parameters) for biomedical question answering and relation extraction using a mixture of clinical and general English text. Whilst these approaches outperformed existing smaller specific purpose models [177, 644] in medical question answering, they currently underperform the larger general purpose LLMs (GPT-3.5/4 and MedPaLM 1/2). However, there remains

debate over whether larger general or specialized clinical models are the best approach. Looking at models up to GPT-3, Lehman et al. [297] question the effectiveness of LLM in-context learning approaches by showing that small specialized clinical models fine-tuned on limited annotated data outperform the former.

Finally, LLMs have also been applied to a range of more specific medical question-answering tasks, including evaluating GPT-3 on its' ability to triage and diagnose cases [301], responding to social media genetics [134] and general [30] patient questions (ChatGPT), answering questions from the Korean general surgery board exams (GPT-3.5, GPT-4) [393], consultation and medical note taking [296], and answering ophthalmology questions [21].

3.7.2 Medical Information Retrieval

Medical text often contains domain-specific abbreviations, acronyms, and technical terms presenting specific information retrieval challenges. This has led LLMs also to be applied to help structure and extract data from medical sources.

Agrawal et al. [10] use InstructGPT (GPT-3) with prompt templates (zero- and one-shot) for clinical information extraction, such as extracting medication dosage and frequency from medical notes or disambiguation of medical acronyms. They also introduce two methods for converting the LLM output into a structured format using a *verbilizer* for mapping to classification labels and a *resolver* for more complex structured outputs such as lists (GPT-3 + R).

Rajkomar et al. [448] take a different approach by treating medical acronym disambiguation as a translation task and training a specialized end-to-end T5 LLM. To preserve privacy, they also use a training dataset generated from public web pages (without medical acronyms) and web-scale reverse substitution of medical acronyms, with only evaluation done on actual clinical notes.

Finally, Gu et al. [178] use GPT-3.5 and knowledge distillation to train a PubMedBERT model for adverse drug event extraction (entity and relation). The distilled PubMedBERT model outperforms GPT-3.5 and GPT-4, and performs similarly to specialized models that use supervised learning.

3.8 Reasoning

Mathematical and algorithmic tasks often require a different set of capabilities than traditional NLP tasks, such as understanding mathematical operations, complex multi-step reasoning, and longer-term planning. Therefore, the applicability of LLMs to these tasks, and methods for improving their capabilities, is an active area of research.

For mathematical reasoning tasks, Uesato et al. [560] test a range of fine-tuning (supervised and RLHF), prompting (zero-shot and few-shot), and re-ranking (majority voting and reward model) to evaluate whether they improve a base LLM's (70B parameters) ability to generate accurate reasoning steps on word-based maths problems in the GSM8K dataset [95]. Whilst fine-tuning on intermediate steps ("process-based") performs similarly to using only final answers ("outcome-based") on final answer correctness, processed-based approaches are found to generate significantly fewer errors in reasoning.

Huang et al. [222] take this a step further by showing that the mathematical reasoning ability of a PaLM LLM on the GSM8K dataset can be *self-improved* through fine-tuning on a dataset of high-confidence reasoning paths generated by the same PaLM base model.

Using only prompting, Kojima et al. [273] find that zero-shot CoT prompting alone significantly improves the performance of GPT-3 and PaLM LLMs over standard zero- and few-shot prompting on the MultiArith and GSM8K datasets. While Li et al. [312] introduce DIVERSE, a prompting approach that uses a diverse set of prompts for each question and a trained verifier (with reasoning step awareness) to improve further GPT-3.5's performance on GSM8K and other reasoning benchmarks. Finally, Shridhar et al. [502] take a novel approach by training new models to break down a mathematical word problem into *Socratic subquestions* to guide the answer of either other LLMs or human learners. GPT-3 prompted with these subquestions outperforms simple one-shot prompting on the GSM8K dataset.

Stolfo et al. [525] evaluate a range of LLMs (including GPT-3) at mathematical reasoning using a new framework to understand the causal impact of different input factors (e.g framing, operands, and operations). Instruction fine-tuned GPT-3 models are found to be significantly more robust and sensitive than the smaller LLMs evaluated.

Other LLM use cases in algorithmic and mathematical reasoning have also been proposed. Gadgil et al. [159] apply a Codex LLM with prompt en-

gineering and filtering to the task of mathematical formalization (in the context of theorem proving). Webb et al. [595] evaluate GPT-3.5's capacity for analogical reasoning using tasks that emulate Raven's Standard Progressive Matrices (SPM), letter string analogies, and verbal analogies. GPT-3.5 is shown to generally outperform human participants (undergraduates) at matrix reasoning and verbal analogies, but with more mixed results on letter string analogies. Yu et al. [654] introduce the ALERT benchmark to evaluate LLM reasoning across ten skills (logistic, causal, commonsense, abductive, spatial, analogical, argument, and deductive reasoning, as well as textual entailment and mathematics). Ruis et al. [464] study LLMs' capability to interpret implicatures, for example, whether they understand the response "I wore gloves" to the question "Did you leave fingerprints?" as meaning "No"; finding that lots of models perform close to random. Finally, Valmeekam et al. [562] propose a new assessment framework for *common-sense* planning and find that existing LLMs GPT-3.5 and BLOOM perform poorly. Using the framework for the Blocksworld domain (planning tasks with different colored blocks on a surface), the best GPT-3.5 model only came up with a valid plan 5% of the time, compared to 78% of human participants.

⚠ Sub-Human-Performance [562, 607]

Existing LLMs struggle to match human performance on reasoning benchmarks.

Another line of work has investigated the intersection of LLMs and causal reasoning [425, 253]. Kıcıman et al. [286] argue that GPT-3.5/4 outperform existing algorithms in three causal benchmarks. In contrast, Gao et al. [164] evaluate ChatGPT on three causal reasoning tasks (distinct from Kıcıman et al. [286]) and find that it performs rather poorly; further, few-shot and chain-of-thought prompting sometimes further exacerbates its performance. Srivastava et al. [519] propose 14 causal reasoning tasks, some of which are considered to be very hard [534]. Similarly, Jin et al. [244] curate another causal inference task and posit that current LLMs still fail to generalize. Lampinen et al. [288] study whether LLMs can generalize causal intervention strategies from few-shot examples. Willig et al. [607] conjecture that current LLMs are "causal parrots", simply reciting causal knowledge embedded in their data rather than doing causal reasoning [253].

Overall, while LLMs show some capacity for more complex reasoning, the relatively poor performance of LLMs on a number of reasoning tasks and benchmarks [562, 164, 244] stands in contrast to the often human level performance being seen in other capabilities [61, 263].

3.9 Robotics and Embodied Agents

LLMs have also started to be incorporated into robotics applications to provide high-level planning and contextual knowledge.

Ahn et al. [14] implement a PaLM-540B LLM in the SayCan architecture to break down high-level text-based instructions into a sequence of lower-level robot tasks that can be executed. The authors use the LLM to propose possible next actions via iteratively scoring the most likely of a defined set of low-level tasks based on the high-level text input. The low-level task to be executed is then determined by combining the low-level tasks proposed by the LLM with affordance functions which determine the probability of the robot completing the task given the current low-level context.

Driess et al. [129] take this concept a step further by combining the PaLM-540B LLM with additional input modalities (22B parameter vision transformer) to create the PaLM-E model. By introducing images into the input, the PaLM-E model can predict which low-level tasks are possible given the current state, whether the previous low-level tasks executed failed, and incorporate images into long-horizon planning, allowing it to outperform the original SayCan results.

Another approach has been to use LLMs to generate code for robotics tasks. Vemprala et al. [564] combine ChatGPT with a pre-defined high-level function library of robotic capabilities for human *on the loop* robotics tasks. By providing details of the function library in the prompt, ChatGPT is then shown to be able to break down high-level natural language instructions into a set of lower-level function calls, which can then be executed on the robot if the human is satisfied it is accurate. This is another example of the *API definition* 13 approach, also used in computer programming [532]. Other related works that use LLMs to generate code for robotics applications include using an LLM for hierarchical code generation to write robot policies (Codex) [316], to generate code policies and main-

tain a written state (GPT-3.5) [647], and using an LLM for code-based task planning (GPT-3, Codex) [510].

Finally, LLMs have also been combined with modality-to-text pre-processing to provide the LLM with additional input from the robot's environment. Liu et al. [338] use GPT-4 as part of the REFLECT framework for detecting and explaining robot failures. To achieve this, multi-modal sensory inputs are first converted into a text-based hierarchical summary at the sensory, event, and sub-goal levels. The hierarchical summary then prompts the LLM to detect and analyze failures. Similarly, Huang et al. [225] combine an LLM (InstructGPT, PaLM) with multiple sources of text-based environment feedback for robotic task planning.

⚠ Single Modality [338, 14, 564]

While LLMs can help robots or agents understand instructions and add high-level planning capabilities, their inability to directly learn from image, audio or other sensor modalities constrain their applications.

For agents in simulated worlds, Wang et al. [579] use the GPT-4 LLM within their VOYAGER framework to create a Minecraft agent that can autonomously explore, acquire new skills and complete tasks. First, they use GPT-4 to propose new tasks for the agent to complete as part of the *automatic curriculum*. Then, they ask it to generate code to solve the proposed task given the current state to add to its *skills library*, which can then be used in the future (similar to the API approach 13 used by Vemprala et al. [564]). Finally, the authors use GPT-4 to verify whether the executed code has achieved the proposed task. This framework outperforms prompting approaches such as ReAct, Reflexion, and AutoGPT (Sec. 2.7).

Prior work using LLMs for planning in simulated worlds include: Wang et al. [591] using GPT-3 for Minecraft, Huang et al. [224] using GPT-3 and Codex in VirtualHome, and Nottingham et al. [389] using Codex for Minecraft.

3.10 Social Sciences & Psychology

The rapid advancements of LLMs have fostered the use of such models across research in the psychological and behavioral sciences. Reviewing the existing literature, we have identified three main areas and tasks in which LLMs have been used in the context of the psychological and behavioral sciences: using LLMs to simulate human behavioral experiments [e.g., 22, 176, 211, 614, 126], analyzing the personality traits of LLMs [e.g., 367, 414, 470], and employing them as artificial agents to model social relationships [409]. See Fig. 15 for an illustration.

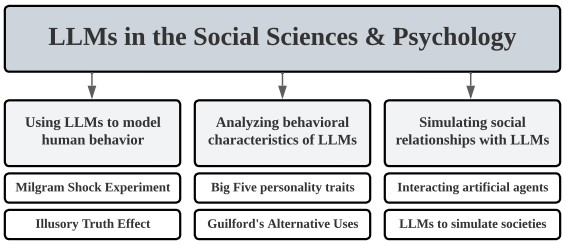


Figure 15: Use cases of LLMs in the social sciences and psychology can mainly be structured into three categories: using LLMs to model human behavior [e.g., 12, 211], analyzing behavioral characteristics of LLMs [e.g., 414], and using LLMs to simulate social relationships [e.g., 408].

3.10.1 Modeling Human Behavior

In the behavioral sciences, there is an increasing interest in using LLMs as models for psychological experiments. Being able to model human behavior computationally through language models would entail a variety of advantages over using human participants: experiments with LLMs are cheaper, faster, can be scaled easier, and are potentially less sensitive to ethical considerations [176]. In light of this, various works have compared LLMs with human participants from a behavioral perspective.

Argyle et al. [22] demonstrate how LLMs can generate responses corresponding to virtual participants in behavioral experiments. They do so by using LLMs to generate samples of responses to studies related to political opinions and voting behavior. In particular, the authors investigate three studies: the first asks participants to list words associated with outgroup partisans, and the second and third focus on vote prediction based on demographics. Across scenarios, experimental results demonstrate that GPT-3 provides answers that closely align with human responses.

Horton [211] argue that LLMs can be used to computationally model human behavior and demonstrate such an ability in economics by exploring their behavior in economic scenarios. They conducted four experiments focusing on economic decision-making using GPT-3, showing that the

LLM can approximately replicate results obtained with human individuals.

Griffin et al. [176] investigate the suitability of LLMs to model psychological change. In their study, the authors assess LLM responses to two behavioral tests, the *illusory truth effect* [ITE; 194] and an experiment measuring the influence of populist news to change in political views [55]. The results demonstrate that in both scenarios, human judgments tend to align with LLM-based judgments, indicating that LLMs have the potential to model the effect of influence on human individuals.

Aher et al. [12] introduce *the Turing Experiment* (TE) to measure an LLM's suitability to model human behavior. A TE consists of inputs to the LLM that signal a certain demographic (e.g., names or occupations) as well as a set of experimental details and corresponding outputs used to simulate human behavior. The authors apply their approach to four individual tests, namely an ultimatum game from behavioral economics [214, 279], garden-path sentences used in psycholinguistics [89, 411], the Milgram Shock Experiment from social psychology [364], and the wisdom of crowds task used to measure collective social intelligence [375]. Demographic details are simulated via gender titles and surnames. The results show that LLMs largely align with human behavior across the tests. However, the authors note that LLM size matters and that larger models tend to provide results that are more aligned with human responses.

Aher et al. [12] point out that the LLMs were most likely exposed to the four behavioral experiments during their pre-training. To account for that, the authors create artificial variations of the experiments with conditions that differ from previous studies. Additionally, the authors note that a potential risk with using LLMs to simulate human responses is the introduction of generations that contain biases stemming from the models' training data.

⚠ Social Biases [12, 367]

Unbalanced views and opinions in the training data skew the LLMs towards biased human behaviors.

Park et al. [409] replicate a set of 8 psychological studies from the Many Labs 2 project [270] using GPT-3 to assess the LLM for its ability to simulate human behavioral data. Such studies include tests in which subjects are asked to choose between a kiss from a favorite movie star and $50 [462] and where subjects had to decide between paying a traffic violation fine and going to court [461]. These experiments show that GPT-3 replicates only 37.5% of the effects obtained from human participants. The authors argue that these results are attributed to humans and LLMs representing inherently different cognitive systems.

Maddela et al. [353] study identifying unhelpful thought patterns and possible reframings to facilitate mental health. They release a dataset called PATTERNREFRAME and evaluate GPT-3.5 on it, showing that it can perform very well without additional training. They conclude that practitioners of cognitive behavioral therapy may benefit from using LLMs to produce richer training material.

3.10.2 Analyzing Behavioral Characteristics of LLMs

In addition to using LLMs as models for human behavior, various existing works study LLMs by analyzing their personality traits.

Jiang et al. [242] do so by introducing the *Machine Personality Inventory* (MPI) dataset, a collection of items to assess personalities according to the Big Five personality factors: extraversion, agreeableness, openness, conscientiousness, and neuroticism [358].

Miotto et al. [367] assess GPT-3's personalities using the HEXACO [27] and Human Values [488] scales. Their experimental results reveal that GPT-3 obtains personality and value scores that align with human participants. Miotto et al. [367] provide an extensive analysis of varying temperature values used to prompt the LLM, finding that an increased temperature yields changes in the model's personalities, e.g., GPT-3 shows a higher unwillingness to manipulate as well as increased scores on anxiety. Similar results were obtained concerning the Human Values scale, where model responses varied substantially for different temperature values.

In line with this work, Pellert et al. [414] argue that LLMs possess psychological traits as observed in human individuals and can be assessed through psychometric tests. The authors conduct experiments measuring, among others, the Big Five personality traits in a zero-shot setup. In contrast, to Miotto et al. [367], Pellert et al. [414] investigate smaller models based on BERT and find that different variants of BERT score across the five personalities in a fairly homogeneous fashion, with

traits that are high on agreeableness and extraversion, but low on neuroticism.

In a related fashion, Stevenson et al. [523] assess LLM performance (GPT-3) on the Guilford's Alternative Uses Test [AUT; 181], a test to assess human creativity. The test asks participants to suggest uses for physical objects (e.g., a book or a fork). Comparing the AUT test performance of GPT-3 to that of psychology students, the authors found that human responses score higher on originality and surprise, whereas GPT-3's responses were more useful.

Kosinski [277] test Theory of Mind (ToM) in LLMs. ToM refers to the ability to track others' unobservable mental states, such as intentions, beliefs, or desires. The authors find that among LLMs of the GPT family, recent models can increasingly solve ToM tasks without having been explicitly trained to do so. For instance, while GPT-2 shows virtually no capability of solving ToM tasks, GPT-3.5 (based on InstructGPT) and GPT-4 performed similarly to 6- and 7-year-old children, respectively. Gandhi et al. [162] present a template-based framework for generating synthetic samples to evaluate ToM in LLMs, which are then applied to five recently developed LLMs (incl. GPT-3, GPT-4, LLaMA, and Claude). The authors show that most models struggle with ToM in its basic forms. However, GPT-4 performs closest to the human comparison of all tested models.

3.10.3 Simulating Social Relationships

While most previous works measure LLMs as models for human behavior through replicating human behavioral studies, Park et al. [408] use the power of LLMs to model the interaction between artificial agents. The authors model a community of 25 artificial agents interacting in a digital environment to achieve this. Each character has unique traits, and the characters interact with each other through natural language. Simulating such societies, the authors observe emergent social behaviors (e.g., forming new relationships and attending events) between agents that are formed without any human interaction.

3.11 Synthetic Data Generation

The ability of LLMs to perform in-context learning allows them to be prompted to generate synthetic datasets for training much smaller domain-specific models.

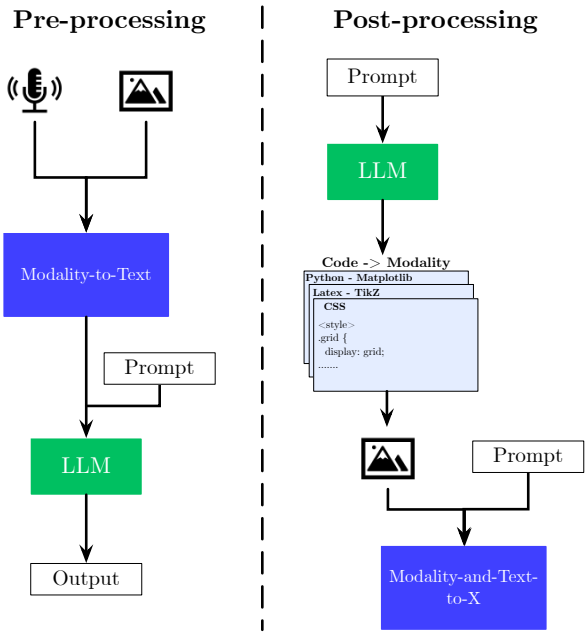


Figure 16: **Modality Conversion**. Illustration of using models with other input modalities as pre or post-processing steps in an LLM pipeline [148, 329, 338, 225, 315]. For some use cases, this approach can be used as an alternative to training a multi-modal model or using a shared embedding space.

Wang et al. [583] propose using GPT-3 to label datasets more cost-effectively than human labelers. These labeled datasets can then be used to train more compute-efficient smaller models. To evaluate this approach, RoBERTa and PEGASUS models are trained for 9 NLP tasks using human and GPT-3 generated labels. GPT-3 labels are shown to outperform human labels when labeling budgets are small, but higher-quality human labels tend to lead to better models at higher labeling budgets.

Similarly, Ding et al. [123] propose three prompting approaches for training data generation with GPT-3: unlabeled data annotation (generate labels for known examples), training data generation (generate examples and labels), and assisted training data generation (with Wikidata provided as additional context). Fine-tuning a smaller BERT model for text classification and NER tasks using these approaches showed results similar to or worse than using GPT-3 directly.

Gunasekar et al. [182] leverage synthetic data generation with GPT-3.5 to train a new code generation LLM (see Sec. 3.3.1). The generated data consists of synthetic Python textbooks focusing on reasoning, basic algorithmic skills, and synthetic Python exercises. One important finding of this

work is that introducing randomness into data generation is crucial, all while ensuring the examples maintain their quality and coherence.

Yoo et al. [648] propose GPT3Mix to generate additional synthetic data from an existing dataset for classification tasks. GPT3Mix uses GPT-3 with a prompt containing real examples from the dataset and a task specification to create synthetic examples and *pseudo-labels* jointly. This new augmented dataset is then used to fine-tune BERT and DistilBERT models. This method combines data augmentation approaches with knowledge distillation by training smaller classification models using soft labels.

Bonifacio et al. [51] propose InPars, a method for using LLMs to generate synthetic retrieval examples for fine-tuning on information retrieval tasks. GPT-3 is few-shot prompted to generate a relevant question for a randomly sampled document along with the question's associated probability. A smaller monoT5 model is then fine-tuned using this dataset to rank relevant documents for a given question. The fine-tuned model outperforms only pre-trained models but performs worse than models fine-tuned using the existing MS MARCO training dataset [32].

Dai et al. [104] introduce AugGPT, which uses ChatGPT (GPT-3.5) to augment each example in a small base dataset with six additional rephrased synthetic examples. This new augmented dataset is then used to fine-tune a specialized BERT model. This approach outperforms existing augmentation approaches, such as word and character substitution.

Finally, instead of generating synthetic data to achieve a specialized task, Shridhar et al. [503] propose Decompositional Distillation, which aims to use synthetic data to replicate in smaller models the multi-step reasoning capabilities, such as CoT, that emerge in larger LLMs. First, GPT-3 is used with a manually designed few-shot prompt to decompose a problem into (sub-question, sub-solution) pairs. This synthetic sub-question dataset is then used to fine-tune a T5 *problem decomposer* to generate sub-questions. Finally, a GPT-2 *problem solver* is fine-tuned to provide the sub-solutions to the teacher-generated sub-questions.

Overall, while LLM-generated synthetic data can potentially bring significant cost benefits, the greater its role, the higher the potential for it to fail to capture the true distribution and potentially lead to model collapse [506].

⚠ Hallucinated Distributions [506]

Using LLMs for fully synthetic data generation is currently constrained by our inability to verify whether the synthetic data generated is representative of the true distribution in the corresponding real-world data.

In cases where the LLM is only used to label existing data [583, 123] this will likely reduce the risk of generating an unrepresentative training distribution (although hallucinated labels remain an issue). Where the LLM is used to generate (or partially generate) both the input and the target [123, 104, 182, 51, 503] the issue of hallucinated distributions becomes potentially significant.

4 Related Work

Closest to ours is the concurrent work by Zhao et al. [673], who provide an extensive survey of large language models and associated topics. Mialon et al. [363] focus on surveying augmented language models, i.e., "language models with reasoning skills and the ability to use tools". Tornede et al. [555] survey LLMs in the context of AutoML methods, highlighting existing methods and challenges in leveraging these for improving LLMs. Tang et al. [539] survey LLM-generated text detection techniques. Chang et al. [72] concurrently survey evaluation tasks of LLMs.

The literature also contains several previous surveys and evaluations specific to individual application domains that reference LLMs, including: chatbots [345], computational biology [558, 217], computer programming [499], medicine [381, 610, 590, 381], law [101, 531], knowledge work [140, 621], and reasoning [223].

5 Conclusion

In this work, we identify several unsolved challenges of large language models, provide an overview of their current applications, and discuss how the former constrain the latter. By highlighting the limitations of existing methods, we hope to foster future research addressing these. We also hope that by providing an overview of the approaches used in different applied areas, we can facilitate the transfer of ideas between domains and target further research.

Acknowledgements

We thank Abhishek Kumar and Stella Rose Biderman for fruitful discussions and feedback on the draft.

References

[1] A blog post detailed a Sam Altman freakout about a huge chips shortage threatening OpenAI. Then it was taken down.

[2] Open LLM Leaderboard - a Hugging Face Space by HuggingFaceH4.

[3] Reproducibility — PyTorch 2.0 documentation.

[4] 2023. Negative prompts for text generation. Section: Prompting.

[5] 2023. Reproducibility. Page Version ID: 1163331755.

[6] A. Abbas, K. Tirumala, D. Simig, S. Ganguli and A. S. Morcos. 2023. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*.

[7] J. D. Abernethy, A. Agarwal, T. V. Marinov and M. K. Warmuth. 2023. A mechanism for sample-efficient in-context learning for sparse retrieval tasks. *ArXiv*, abs/2305.17040.

[8] D. Adiwardana, M.-T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

[9] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville and M. Bellemare. 2021. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc.

[10] M. Agrawal, S. Hegselmann, H. Lang, Y. Kim and D. Sontag. 2022. Large language models are zero-shot clinical information extractors. *arXiv preprint arXiv:2205.12689*.

[11] P. Agrawal, C. Alberti, F. Huot, J. Maynez, J. Ma, S. Ruder, K. Ganchev, D. Das et al. 2022. Qameleon: Multilingual qa with only 5 examples. *arXiv preprint arXiv:2211.08264*.

[12] G. Aher, R. I. Arriaga and A. T. Kalai. 2022. Using large language models to simulate multiple humans. *arXiv preprint arXiv:2208.10264*.

[13] O. Ahia, S. Kumar, H. Gonen, J. Kasai, D. R. Mortensen, N. A. Smith and Y. Tsvetkov. 2023. Do all languages cost the same? tokenization in the era of commercial language models. *arXiv preprint arXiv:2305.13707*.

[14] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan et al. 2022. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*.

[15] J. Ainslie, T. Lei, M. de Jong, S. Ontañón, S. Brahma, Y. Zemlyanskiy, D. Uthus, M. Guo et al. 2023. Colt5: Faster long-range transformers with conditional computation. *arXiv preprint arXiv:2303.09752*.

[16] E. Akyürek, D. Schuurmans, J. Andreas, T. Ma and D. Zhou. 2023. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*.

[17] L. B. Allal, R. Li, D. Kocetkov, C. Mou, C. Akiki, C. M. Ferrandis, N. Muennighoff, M. Mishra et al. 2023. Santacoder: don't reach for the stars!

[18] J. Andreas. 2022. Language models as agent models.

[19] C. Anil, Y. Wu, A. Andreassen, A. Lewkowycz, V. Misra, V. Ramasesh, A. Slone, G. Gur-Ari et al. 2022. Exploring Length Generalization in Large Language Models. ArXiv:2207.04901 [cs].

[20] R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

[21] F. Antaki, S. Touma, D. Milad, J. El-Khoury and R. Duval. 2023. Evaluating the performance of chatgpt in ophthalmology: An analysis of its successes and shortcomings. *medRxiv*.

[22] L. P. Argyle, E. C. Busby, N. Fulda, J. Gubler, C. Rytting and D. Wingate. 2022. Out of one, many: Using language models to simulate human samples. *arXiv preprint arXiv:2209.06899*.

[23] V. Aribandi, Y. Tay, T. Schuster, J. Rao, H. S. Zheng, S. V. Mehta, H. Zhuang, V. Q. Tran et al. 2022. Ext5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*.

[24] S. Arora, A. Narayan, M. F. Chen, L. Orr, N. Guha, K. Bhatia, I. Chami, F. Sala et al. 2022. Ask me anything: A simple strategy for prompting language models.

[25] A. Asai, T. Schick, P. Lewis, X. Chen, G. Izacard, S. Riedel, H. Hajishirzi and W.-t. Yih. 2022. Task-aware retrieval with instructions.

[26] N. Asher, S. Bhar, A. Chaturvedi, J. Hunter and S. Paul. 2023. Limits for Learning with Language Models. ArXiv:2306.12213 [cs].

[27] M. C. Ashton and K. Lee. 2009. The hexaco–60: A short measure of the major dimensions of personality. *Journal of personality assessment*, 91(4):340–345.

[28] J. Austin, A. Odena, M. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. Cai et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

[29] AUTOMATIC1111. 2023. Stable Diffusion web UI. Original-date: 2022-08-22T14:05:26Z.

[30] J. W. Ayers, A. Poliak, M. Dredze, E. C. Leas, Z. Zhu, J. B. Kelley, D. J. Faix, A. M. Goodman et al. 2023. Comparing physician and artificial intelligence chatbot responses to patient questions posted to a public social media forum. *JAMA internal medicine*.

[31] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

[32] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara et al. 2018. Ms marco: A human generated machine reading comprehension dataset.

[33] P. Bajaj, C. Xiong, G. Ke, X. Liu, D. He, S. Tiwary, T.-Y. Liu, P. Bennett et al. 2022. Metro: Efficient denoising pretraining of large scale autoencoding language models with model generated signals. *arXiv preprint arXiv:2204.06644*.

[34] A. Bakhtin, S. Gross, M. Ott, Y. Deng, M. Ranzato and A. Szlam. 2019. Real or Fake? Learning to Discriminate Machine from Human Generated Text. ArXiv:1906.03351 [cs, stat].

[35] R. Balestriero, J. Pesenti and Y. LeCun. 2021. Learning in high dimension always amounts to extrapolation. *arXiv preprint arXiv:2110.09485*.

[36] J. Bandy and N. Vincent. 2021. Addressing "documentation debt" in machine learning research: A retrospective datasheet for bookcorpus.

[37] P. Barham, A. Chowdhery, J. Dean, S. Ghemawat, S. Hand, D. Hurt, M. Isard, H. Lim et al. 2022. Pathways: Asynchronous distributed dataflow for ml. *Proceedings of Machine Learning and Systems*, 4:430–449.

[38] M. Bavarian, H. Jun, N. Tezak, J. Schulman, C. McLeavey, J. Tworek and M. Chen. 2022. Efficient training of language models to fill in the middle. *arXiv preprint arXiv:2207.14255*.

[39] N. Belrose, Z. Furman, L. Smith, D. Halawi, I. Ostrovsky, L. McKinney, S. Biderman and J. Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens.

[40] E. Ben Zaken, Y. Goldberg and S. Ravfogel. 2022. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–9, Dublin, Ireland. Association for Computational Linguistics.

[41] S. Biderman, K. Bicheno and L. Gao. 2022. Datasheet for the pile. *arXiv preprint arXiv:2201.07311*.

[42] S. Biderman, U. S. Prashanth, L. Sutawika, H. Schoelkopf, Q. Anthony, S. Purohit and E. Raff. 2023. Emergent and Predictable Memorization in Large Language Models. ArXiv:2304.11158 [cs].

[43] S. Biderman and W. J. Scheirer. 2021. Pitfalls in machine learning research: Reexamining the development cycle.

[44] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit et al. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

[45] S. R. Biderman. 2023. [...] we aren't running out of text data any time soon. ml researchers massively underestimate how much text is out there. https://twitter.com/BlancheMinerva/status/1644154144431677442?s=20. Accessed: 2023-05-28.

[46] A. Birhane, V. U. Prabhu and E. Kahembwe. 2021. Multimodal datasets: misogyny, pornography, and malignant stereotypes. *arXiv preprint arXiv:2110.01963*.

[47] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy et al. 2022. Gpt-neox-20b: An open-source autoregressive language model.

[48] A. Blair-Stanek, N. Holzenberger and B. Van Durme. 2023. Can gpt-3 perform statutory reasoning? *arXiv preprint arXiv:2302.06100*.

[49] J. Bommarito, M. Bommarito, D. M. Katz and J. Katz. 2023. Gpt as knowledge worker: A zero-shot evaluation of (ai) cpa capabilities. *arXiv preprint arXiv:2301.04408*.

[50] M. Bommarito II and D. M. Katz. 2022. Gpt takes the bar exam. *arXiv preprint arXiv:2212.14402*.

[51] L. Bonifacio, H. Abonizio, M. Fadaee and R. Nogueira. 2022. Inpars: Unsupervised dataset generation for information retrieval. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2387–2392, New York, NY, USA. Association for Computing Machinery.

[52] S. Borgeaud, A. Mensch, J. Hoffmann, T. Cai, E. Rutherford, K. Millican, G. v. d. Driessche, J.-B. Lespiau et al. 2021. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*.

[53] A. Borji. 2023. A Categorical Archive of ChatGPT Failures. ArXiv:2302.03494 [cs].

[54] A. Borzunov, D. Baranchuk, T. Dettmers, M. Ryabinin, Y. Belkada, A. Chumachenko, P. Samygin and C. Raffel. 2022. Petals: Collaborative inference and fine-tuning of large models. *arXiv preprint arXiv:2209.01188*.

[55] L. Bos, C. Schemer, N. Corbu, M. Hameleers, I. Andreadis, A. Schulz, D. Schmuck, C. Reinemann et al. 2020. The effects of populism as a social identity frame on persuasion and mobilisation: Evidence from a 15-country experiment. *European Journal of Political Research*, 59(1):3–24.

[56] D. Britz, M. Y. Guan and M.-T. Luong. 2017. Efficient attention using a fixed-size memory representation. *arXiv preprint arXiv:1707.00110*.

[57] A. Z. Broder, M. Charikar, A. M. Frieze and M. Mitzenmacher. 1998. Min-wise independent permutations. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 327–336.

[58] G. Brown, M. Bun, V. Feldman, A. Smith and K. Talwar. 2021. When is memorization of irrelevant training data necessary for high-accuracy learning? In *Proceedings of the 53rd annual ACM SIGACT symposium on theory of computing*, pages 123–132.

[59] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

[60] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre et al. 2018. The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228*.

[61] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

[62] C. Burns, H. Ye, D. Klein and J. Steinhardt. 2022. Discovering latent knowledge in language models without supervision.

[63] A. Calderwood, N. Wardrip-Fruin and M. Mateas. 2022. Spinning coherent interactive fiction through foundation model prompts. *International Conference of Computation and Creativity*.

[64] N. Carlini, M. Jagielski, C. A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas et al. 2023. Poisoning Web-Scale Training Datasets is Practical. ArXiv:2302.10149 [cs].

[65] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos and D. Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *USENIX Security Symposium*, volume 267.

[66] N. Carlini, M. Nasr, C. A. Choquette-Choo, M. Jagielski, I. Gao, A. Awadalla, P. W. Koh, D. Ippolito et al. 2023. Are aligned neural networks adversarially aligned?

[67] N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown et al. 2020. Extracting training data from large language models.

[68] S. Casper, J. Lin, J. Kwon, G. Culp and D. Hadfield-Menell. 2023. Explore, establish, exploit: Red teaming language models from scratch. *arXiv preprint arXiv:2306.09442*.

[69] T. Chakrabarty, V. Padmakumar and H. He. 2022. Help me write a poem: Instruction tuning as a vehicle for collaborative poetry writing. *arXiv preprint arXiv:2210.13669*.

[70] I. Chalkidis, I. Androutsopoulos and N. Aletras. 2019. Neural legal judgment prediction in english. *arXiv preprint arXiv:1906.02059*.

[71] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras and I. Androutsopoulos. 2020. Legal-bert: The muppets straight out of law school. *arXiv preprint arXiv:2010.02559*.

[72] Y. Chang, X. Wang, J. Wang, Y. Wu, K. Zhu, H. Chen, L. Yang, X. Yi et al. 2023. A Survey on Evaluation of Large Language Models. ArXiv:2307.03109 [cs].

[73] B. Chen, X. Cheng, L. ao Gengyang, S. Li, X. Zeng, B. Wang, G. Jing, C. Liu et al. 2023. xtrimopglm: Unified 100b-scale pre-trained transformer for deciphering the language of protein. *bioRxiv*.

[74] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre and J. Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

[75] L. Chen, M. Zaharia and J. Zou. 2023. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. ArXiv:2305.05176 [cs].

[76] L. Chen, M. Zaharia and J. Zou. 2023. How is ChatGPT's behavior changing over time? ArXiv:2307.09009 [cs].

[77] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda et al. 2021. Evaluating large language models trained on code.

[78] M. Chen, A. Papangelis, C. Tao, S. Kim, A. Rosenbaum, Y. Liu, Z. Yu and D. Hakkani-Tur. 2023. Places: Prompting language models for social conversation synthesis. *arXiv preprint arXiv:2302.03269*.

[79] S. Chen, S. Wong, L. Chen and Y. Tian. 2023. Extending context window of large language models via positional interpolation.

[80] T. Chen, Z. Zhang, A. Jaiswal, S. Liu and Z. Wang. 2023. Sparse moe as the new dropout: Scaling dense and self-slimmable transformers.

[81] X. Chen, M. Lin, N. Schärli and D. Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

[82] L. Cheng, X. Li and L. Bing. 2023. Is gpt-4 a good data analyst?

[83] D. Choe, R. Al-Rfou, M. Guo, H. Lee and N. Constant. 2019. Bridging the Gap for Tokenizer-Free Language Models. ArXiv:1908.10322 [cs].

[84] J. H. Choi, K. E. Hickman, A. Monahan and D. Schwarcz. 2023. Chatgpt goes to law school. *Available at SSRN*.

[85] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis et al. 2020. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*.

[86] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

[87] M. Christ, S. Gunn and O. Zamir. 2023. Undetectable Watermarks for Language Models.

[88] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg and D. Amodei. 2017. Deep reinforcement learning from human preferences.

[89] K. Christianson, A. Hollingworth, J. F. Halliwell and F. Ferreira. 2001. Thematic roles assigned along the garden path linger. *Cognitive psychology*, 42(4):368–407.

[90] H. W. Chung. 2023. Missing model details (tweet).

[91] H. W. Chung, X. Garcia, A. Roberts, Y. Tay, O. Firat, S. Narang and N. Constant. 2023. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. In *The Eleventh International Conference on Learning Representations*.

[92] H. W. Chung, D. Garrette, K. C. Tan and J. Riesa. 2020. Improving multilingual models with language-clustered vocabularies. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4536–4546, Online. Association for Computational Linguistics.

[93] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang et al. 2022. Scaling instruction-finetuned language models.

[94] J. H. Clark, D. Garrette, I. Turc and J. Wieting. 2022. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91.

[95] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek et al. 2021. Training verifiers to solve math word problems.

[96] D. Cohen, M. Ryu, Y. Chow, O. Keller, I. Greenberg, A. Hassidim, M. Fink, Y. Matias et al. 2022. Dynamic planning in open-ended dialogue using reinforcement learning. *arXiv preprint arXiv:2208.02294*.

[97] R. Cohen, M. Hamri, M. Geva and A. Globerson. 2023. LM vs LM: Detecting Factual Errors via Cross Examination. ArXiv:2305.13281 [cs].

[98] T. Computer. 2023. Redpajama: An open source recipe to reproduce llama training dataset.

[99] A. Conmy, A. N. Mavor-Parker, A. Lynch, S. Heimersheim and A. Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *arXiv preprint arXiv:2304.14997*.

[100] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott et al. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

[101] A. B. Cyphert. 2021. A human being wrote this law review article: Gpt-3 and the practice of law. *UC Davis L. Rev.*, 55:401.

[102] D. Dai, L. Dong, Y. Hao, Z. Sui, B. Chang and F. Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

[103] D. Dai, Y. Sun, L. Dong, Y. Hao, Z. Sui and F. Wei. 2022. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*.

[104] H. Dai, Z. Liu, W. Liao, X. Huang, Z. Wu, L. Zhao, W. Liu, N. Liu et al. 2023. Chataug: Leveraging chatgpt for text data augmentation.

[105] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le and R. Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

[106] H. Dalla-Torre, L. Gonzalez, J. Mendoza Revilla, N. Lopez Carranza, A. Henryk Grywaczewski, F. Oteri, C. Dallago, E. Trop et al. 2023. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, pages 2023–01.

[107] T. Dao, D. Y. Fu, S. Ermon, A. Rudra and C. Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*.

[108] T. Dao, D. Y. Fu, K. K. Saab, A. W. Thomas, A. Rudra and C. Ré. 2023. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. ArXiv:2212.14052 [cs].

[109] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski and R. Liu. 2020. Plug and play language models: A simple approach to controlled text generation.

[110] J. Dauparas, I. Anishchenko, N. Bennett, H. Bai, R. J. Ragotte, L. F. Milles, B. I. M. Wicky, A. Courbet et al. 2022. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56.

[111] N. De Cao, W. Aziz and I. Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[112] M. Dehghani, A. Arnab, L. Beyer, A. Vaswani and Y. Tay. 2022. The Efficiency Misnomer. ArXiv:2110.12894 [cs, stat].

[113] M. Dehghani, Y. Tay, A. A. Gritsenko, Z. Zhao, N. Houlsby, F. Diaz, D. Metzler and O. Vinyals. 2021. The benchmark lottery. *arXiv preprint arXiv:2107.07002*.

[114] L. Del Corro, A. Del Giorno, S. Agarwal, B. Yu, A. Awadallah and S. Mukherjee. 2023. SkipDecode: Autoregressive Skip Decoding with Batching and Caching for Efficient LLM Inference. ArXiv:2307.02628 [cs].

[115] A. Deroy, K. Ghosh and S. Ghosh. 2023. How ready are pre-trained abstractive models and llms for legal case judgement summarization?

[116] A. Deshpande, V. Murahari, T. Rajpurohit, A. Kalyan and K. Narasimhan. 2023. Toxicity in chatgpt: Analyzing persona-assigned language models. *arXiv preprint arXiv:2304.05335*.

[117] T. Dettmers, M. Lewis, Y. Belkada and L. Zettlemoyer. 2022. Llm.int8(): 8-bit matrix multiplication for transformers at scale.

[118] T. Dettmers, A. Pagnoni, A. Holtzman and L. Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. ArXiv:2305.14314 [cs].

[119] T. Dettmers, R. Svirschevski, V. Egiazarian, D. Kuznedelev, E. Frantar, S. Ashkboos, A. Borzunov, T. Hoefler et al. 2023. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*.

[120] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

[121] N. Dey, G. Gosal, Zhiming, Chen, H. Khachane, W. Marshall, R. Pathria, M. Tom et al. 2023. Cerebras-gpt: Open compute-optimal language models trained on the cerebras wafer-scale cluster.

[122] S. Diao, X. Li, Y. Lin, Z. Huang and T. Zhang. 2022. Black-box prompt learning for pre-trained language models. *arXiv preprint arXiv:2201.08531*.

[123] B. Ding, C. Qin, L. Liu, L. Bing, S. Joty and B. Li. 2022. Is gpt-3 a good data annotator? *arXiv preprint arXiv:2212.10450*.

[124] J. Ding, S. Ma, L. Dong, X. Zhang, S. Huang, W. Wang and F. Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens.

[125] J. Dodge, M. Sap, A. Marasović, W. Agnew, G. Ilharco, D. Groeneveld, M. Mitchell and M. Gardner. 2021. Documenting large webtext corpora: A case study on the colossal clean crawled corpus. *arXiv preprint arXiv:2104.08758*.

[126] R. Dominguez-Olmedo, M. Hardt and C. Mendler-Dünner. 2023. Questioning the survey responses of large language models. *arXiv preprint arXiv:2306.07951*.

[127] Q. Dong, D. Dai, Y. Song, J. Xu, Z. Sui and L. Li. 2022. Calibrating factual knowledge in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5937–5947, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[128] D. R. Dowty, R. Wall and S. Peters. 2012. *Introduction to Montague semantics*, volume 11. Springer Science & Business Media.

[129] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

[130] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou et al. 2022. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR.

[131] Y. Du, S. Li, A. Torralba, J. B. Tenenbaum and I. Mordatch. 2023. Improving Factuality and Reasoning in Language Models through Multiagent Debate. ArXiv:2305.14325 [cs].

[132] Z. Du, Y. Qian, X. Liu, M. Ding, J. Qiu, Z. Yang and J. Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, Dublin, Ireland. Association for Computational Linguistics.

[133] A. Dunn, J. Dagdelen, N. Walker, S. Lee, A. S. Rosen, G. Ceder, K. Persson and A. Jain. 2022. Structured information extraction from complex scientific text with fine-tuned large language models. *arXiv preprint arXiv:2212.05238*.

[134] D. Duong and B. D. Solomon. 2023. Analysis of large-language model versus human performance for genetics questions. *European Journal of Human Genetics*, pages 1–3.

[135] N. Dziri, X. Lu, M. Sclar, X. L. Li, L. Jiang, B. Y. Lin, P. West, C. Bhagavatula et al. 2023. Faith and Fate: Limits of Transformers on Compositionality. ArXiv:2305.18654 [cs].

[136] N. Dziri, A. Madotto, O. Zaiane and A. J. Bose. 2021. Neural Path Hunter: Reducing Hallucination in Dialogue Systems via Path Grounding. ArXiv:2104.08455 [cs].

[137] E.-M. El-Mhamdi, S. Farhadkhani, R. Guerraoui, N. Gupta, L.-N. Hoang, R. Pinot, S. Rouault and J. Stephan. 2023. On the Impossible Safety of Large AI Models. ArXiv:2209.15259 [cs].

[138] N. Elhage, N. Nanda, C. Olsson, T. Henighan, N. Joseph, B. Mann, A. Askell, Y. Bai et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

[139] A. Elnaggar, M. Heinzinger, C. Dallago, G. Rihawi, Y. Wang, L. Jones, T. Gibbs, T. Feher et al. 2020. Prottrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing. *arXiv preprint arXiv:2007.06225*.

[140] T. Eloundou, S. Manning, P. Mishkin and D. Rock. 2023. Gpts are gpts: An early look at the labor market impact potential of large language models.

[141] F. Faal, K. Schmitt and J. Y. Yu. 2023. Reward modeling for mitigating toxicity in transformer-based language models. *Applied Intelligence*, 53(7):8421–8435.

[142] A. Fan, C. Gardent, C. Braud and A. Bordes. 2021. Augmenting transformers with KNN-based composite memory for dialog. *Transactions of the Association for Computational Linguistics*, 9:82–99.

[143] A. Fan, E. Grave and A. Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

[144] M. Fathi, J. Pilault, P.-L. Bacon, C. Pal, O. Firat and R. Goroshin. 2023. Block-State Transformer. ArXiv:2306.09539 [cs].

[145] W. Fedus, B. Zoph and N. Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

[146] V. Feldman. 2020. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959.

[147] S. Feng, C. Y. Park, Y. Liu and Y. Tsvetkov. 2023. From Pretraining Data to Language Models to Downstream Tasks: Tracking the Trails of Political Biases Leading to Unfair NLP Models. ArXiv:2305.08283 [cs].

[148] W. Feng, W. Zhu, T.-j. Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang et al. 2023. LayoutGPT: Compositional Visual Planning and Generation with Large Language Models. ArXiv:2305.15393 [cs].

[149] E. Ferrara. 2023. Should chatgpt be biased? challenges and risks of bias in large language models. *arXiv preprint arXiv:2304.03738*.

[150] A. Ficek, F. Liu and N. Collier. 2022. How to tackle an emerging topic? combining strong and weak labels for covid news NER. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 488–496, Online only. Association for Computational Linguistics.

[151] C. Fourrier, N. Habib, J. Launay and T. Wolf. 2023. What's going on with the open llm leaderboard? Available from: https://huggingface.co/blog/evaluating-mmlu-leaderboard. Accessed: 27/06/2023.

[152] E. Frantar and D. Alistarh. 2023. Massive language models can be accurately pruned in one-shot. *arXiv preprint arXiv:2301.00774*.

[153] E. Frantar, S. Ashkboos, T. Hoefler and D. Alistarh. 2022. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

[154] D. Fried, A. Aghajanyan, J. Lin, S. Wang, E. Wallace, F. Shi, R. Zhong, W.-t. Yih et al. 2022. Incoder: A generative model for code infilling and synthesis.

[155] A. Frömmgen and L. Kharatyan. 2023. Resolving code review comments with ml. Available from: https://ai.googleblog.com/2023/05/resolving-code-review-comments-with-ml.html. Accessed: 26/06/2023.

[156] J. Fu, S.-K. Ng, Z. Jiang and P. Liu. 2023. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*.

[157] T. Fujii, K. Shibata, A. Yamaguchi, T. Morishita and Y. Sogawa. 2023. How do different tokenizers perform on downstream tasks in scriptio continua languages?: A case study in japanese. *arXiv preprint arXiv:2306.09572*.

[158] I. Gabriel. 2020. Artificial intelligence, values, and alignment. *Minds and machines*, 30(3):411–437.

[159] S. Gadgil, A. R. Tadipatri, A. Agrawal, A. Narayanan and N. Goyal. 2022. Towards automating formalisation of theorem statements using large language models. *36th Conference on Neural Information Processing Systems (NeurIPS 2022) Workshop on MATH-AI*.

[160] T. Gale, D. Narayanan, C. Young and M. Zaharia. 2022. Megablocks: Efficient sparse training with mixture-of-experts. *arXiv preprint arXiv:2211.15841*.

[161] T. Gale, M. Zaharia, C. Young and E. Elsen. 2020. Sparse GPU Kernels for Deep Learning. ArXiv:2006.10901 [cs, stat].

[162] K. Gandhi, J.-P. Fränken, T. Gerstenbrg and N. D. Goodman. 2023. Understanding social reasoning in language models with language models. *arXiv preprint arXiv:2306.15448*.

[163] D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. Bai, S. Kadavath, B. Mann, E. Perez et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*.

[164] J. Gao, X. Ding, B. Qin and T. Liu. 2023. Is chatgpt a good causal reasoner? a comprehensive evaluation. *arXiv preprint arXiv:2305.07375*.

[165] L. Gao, S. Biderman, S. Black, L. Golding, T. Hoppe, C. Foster, J. Phang, H. He et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

[166] L. Gao, J. Tow, S. Biderman, S. Black, A. DiPofi, C. Foster, L. Golding, J. Hsu et al. 2021. A framework for few-shot language model evaluation.

[167] S. Gehman, S. Gururangan, M. Sap, Y. Choi and N. A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462*.

[168] S. Gehrmann, H. Strobelt and A. M. Rush. 2019. GLTR: Statistical Detection and Visualization of Generated Text. ArXiv:1906.04043 [cs].

[169] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge and F. A. Wichmann. 2020. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

[170] A. Glaese, N. McAleese, M. Trębacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger et al. 2022. Improving alignment of dialogue agents via targeted human judgements.

[171] D. Goldberg. 1991. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48.

[172] A. N. Gomez, O. Key, K. Perlin, S. Gou, N. Frosst, J. Dean and Y. Gal. 2022. Interlocking backpropagation: Improving depthwise model-parallelism. *The Journal of Machine Learning Research*, 23(1):7714–7741.

[173] L. Gong, D. He, Z. Li, T. Qin, L. Wang and T. Liu. 2019. Efficient training of BERT by progressively stacking. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2337–2346. PMLR.

[174] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan and W. Chen. 2023. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*.

[175] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz and M. Fritz. 2023. More than you've asked for: A comprehensive analysis of novel prompt injection threats to application-integrated large language models. *arXiv preprint arXiv:2302.12173*.

[176] L. D. Griffin, B. Kleinberg, M. Mozes, K. T. Mai, M. Vau, M. Caldwell and A. Marvor-Parker. 2023. Susceptibility to influence of large language models. *arXiv preprint arXiv:2303.06074*.

[177] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao et al. 2021. Domain-specific language model pretraining for biomedical natural language processing. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–23.

[178] Y. Gu, S. Zhang, N. Usuyama, Y. Woldesenbet, C. Wong, P. Sanapathi, M. Wei, N. Valluri et al. 2023. Distilling large language models for biomedical knowledge extraction: A case study on adverse drug events.

[179] Y. Gu, X. Han, Z. Liu and M. Huang. 2022. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland. Association for Computational Linguistics.

[180] A. Gudibande, E. Wallace, C. Snell, X. Geng, H. Liu, P. Abbeel, S. Levine and D. Song. 2023. The false promise of imitating proprietary llms. *arXiv preprint arXiv:2305.15717*.

[181] J. P. Guilford. 1967. Creativity: Yesterday, today and tomorrow. *The Journal of Creative Behavior*, 1(1):3–14.

[182] S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. D. Giorno, S. Gopi, M. Javaheripi, P. Kauffmann et al. 2023. Textbooks are all you need.

[183] M. Guo, J. Ainslie, D. Uthus, S. Ontanon, J. Ni, Y.-H. Sung and Y. Yang. 2022. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States. Association for Computational Linguistics.

[184] A. Gupta. 2023. Probing Quantifier Comprehension in Large Language Models. ArXiv:2306.07384 [cs].

[185] T. Gupta and A. Kembhavi. 2022. Visual programming: Compositional visual reasoning without training.

[186] K. Guu, K. Lee, Z. Tung, P. Pasupat and M. Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

[187] J. Haase and P. H. P. Hanel. 2023. Artificial muses: Generative artificial intelligence chatbots have risen to human-level creativity.

[188] M. Hahn and N. Goyal. 2023. A theory of emergent in-context learning as implicit structure induction. *ArXiv*, abs/2303.07971.

[189] S. Hamilton. 2023. Blind judgement: Agent-based supreme court modelling with gpt. *arXiv preprint arXiv:2301.05327*.

[190] C. Han, Z. Wang, H. Zhao and H. Ji. 2023. In-context learning of large language models explained as kernel regression. *ArXiv*, abs/2305.12766.

[191] T. Hartvigsen, S. Sankaranarayanan, H. Palangi, Y. Kim and M. Ghassemi. 2022. Aging with grace: Lifelong model editing with discrete key-value adaptors. *arXiv preprint arXiv:2211.11031*.

[192] A. Haviv, O. Ram, O. Press, P. Izsak and O. Levy. 2022. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[193] J. Hazell. 2023. Large language models can be used to effectively scale spear phishing campaigns. *arXiv preprint arXiv:2305.06972*.

[194] E. L. Henderson, S. J. Westwood and D. J. Simons. 2022. A reproducible systematic map of research on the illusory truth effect. *Psychonomic Bulletin & Review*, pages 1–24.

[195] P. Henderson, M. S. Krass, L. Zheng, N. Guha, C. D. Manning, D. Jurafsky and D. E. Ho. 2022. Pile of law: Learning responsible data filtering from the law and a 256GB open-source legal dataset. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[196] D. Hendrycks, C. Burns, S. Basart, A. Critch, J. Li, D. Song and J. Steinhardt. 2020. Aligning ai with shared human values. *arXiv preprint arXiv:2008.02275*.

[197] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song and J. Steinhardt. 2021. Measuring massive multitask language understanding.

[198] D. Hendrycks, N. Carlini, J. Schulman and J. Steinhardt. 2021. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*.

[199] D. Hendrycks and M. Mazeika. 2022. X-risk analysis for ai research. *arXiv preprint arXiv:2206.05862*.

[200] D. Hernandez, T. Brown, T. Conerly, N. DasSarma, D. Drain, S. El-Showk, N. Elhage, Z. Hatfield-Dodds et al. 2022. Scaling laws and interpretability of learning from repeated data. *arXiv preprint arXiv:2205.10487*.

[201] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali et al. 2017. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*.

[202] B. L. Hie, V. R. Shanker, D. Xu, T. U. Bruun, P. A. Weidenbacher, S. Tang, W. Wu, J. E. Pak et al. 2023. Efficient evolution of human antibodies from general protein language models. *Nature Biotechnology*.

[203] P. Hingston and M. Preuss. 2011. Red teaming with coevolution. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 1155–1163. IEEE.

[204] J. Ho and T. Salimans. 2022. Classifier-free diffusion guidance.

[205] J. Hoelscher-Obermaier, J. Persson, E. Kran, I. Konstas and F. Barez. 2023. Detecting Edit Failures In Large Language Models: An Improved Specificity Benchmark. ArXiv:2305.17553 [cs].

[206] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. de las Casas, L. A. Hendricks et al. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*.

[207] A. Holtzman, J. Buys, L. Du, M. Forbes and Y. Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

[208] N. Holzenberger, A. Blair-Stanek and B. Van Durme. 2020. A dataset for statutory reasoning in tax law entailment and question answering. *arXiv preprint arXiv:2005.05257*.

[209] O. Honovich, T. Scialom, O. Levy and T. Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*.

[210] S. Hooker. 2021. The hardware lottery. *Communications of the ACM*, 64(12):58–65.

[211] J. J. Horton. 2023. Large language models as simulated economic agents: What can we learn from homo silicus? *arXiv preprint arXiv:2301.07543*.

[212] M. Horton, S. Mehta, A. Farhadi and M. Rastegari. 2023. Bytes Are All You Need: Transformers Operating Directly On File Bytes. ArXiv:2306.00238 [cs].

[213] N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan and S. Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

[214] D. Houser and K. McCabe. 2014. Experimental economics and experimental game theory. In *Neuroeconomics*, pages 19–34. Elsevier.

[215] J. Howard and S. Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

[216] S. Hsiao. 2023. What's ahead for bard: More global, more visual, more integrated. Available from: https://blog.google/technology/ai/google-bard-updates-io-2023/. Accessed: 28/06/2023.

[217] B. Hu, J. Xia, J. Zheng, C. Tan, Y. Huang, Y. Xu and S. Z. Li. 2022. Protein language models and structure prediction: Connection and progression.

[218] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang and W. Chen. 2021. Lora: Low-rank adaptation of large language models.

[219] Z. Hu, Y. Lan, L. Wang, W. Xu, E.-P. Lim, R. K.-W. Lee, L. Bing and S. Poria. 2023. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

[220] W. Hua, Z. Dai, H. Liu and Q. Le. 2022. Transformer Quality in Linear Time. In *Proceedings of the 39th International Conference on Machine Learning*, pages 9099–9117. PMLR. ISSN: 2640-3498.

[221] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman et al. 2019. Music transformer. In *International Conference on Learning Representations*.

[222] J. Huang, S. S. Gu, L. Hou, Y. Wu, X. Wang, H. Yu and J. Han. 2022. Large language models can self-improve.

[223] J. Huang and K. C.-C. Chang. 2023. Towards Reasoning in Large Language Models: A Survey. ArXiv:2212.10403 [cs].

[224] W. Huang, P. Abbeel, D. Pathak and I. Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

[225] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson et al. 2022. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

[226] Y. Huang, Y. Cheng, A. Bapna, O. Firat, M. X. Chen, D. Chen, H. Lee, J. Ngiam et al. 2018. Gpipe: Efficient training of giant neural networks using pipeline parallelism.

[227] Z. Huang, Y. Shen, X. Zhang, J. Zhou, W. Rong and Z. Xiong. 2023. Transformer-patcher: One mistake worth one neuron. In *The Eleventh International Conference on Learning Representations*.

[228] I. Hubara, B. Chmiel, M. Island, R. Banner, J. Naor and D. Soudry. 2021. Accelerated sparse neural training: A provable and efficient method to find n:m transposable masks. In *Advances in Neural Information Processing Systems*, volume 34, pages 21099–21111. Curran Associates, Inc.

[229] HuggingFace. 2023. Huggingchat v0.3.0. Available from: https://huggingface.co/chat. Accessed: 28/06/2023.

[230] C. Hwang, W. Cui, Y. Xiong, Z. Yang, Z. Liu, H. Hu, Z. Wang, R. Salas et al. 2022. Tutel: Adaptive mixture-of-experts at scale. *arXiv preprint arXiv:2206.03382*.

[231] J. P. A. Ioannidis. 2005. Why Most Published Research Findings Are False. *PLoS Medicine*, 2(8):e124.

[232] D. Ippolito, A. Yuan, A. Coenen and S. Burnam. 2022. Creative writing with an ai-powered writing assistant: Perspectives from professional writers. *arXiv preprint arXiv:2211.05030*.

[233] G. Irving, P. Christiano and D. Amodei. 2018. Ai safety via debate. *arXiv preprint arXiv:1805.00899*.

[234] K. Y. Iu and V. M.-Y. Wong. 2023. Chatgpt by openai: The end of litigation lawyers? *Available at SSRN*.

[235] S. Iyer, X. V. Lin, R. Pasunuru, T. Mihaylov, D. Simig, P. Yu, K. Shuster, T. Wang et al. 2022. Opt-iml: Scaling language model instruction meta learning through the lens of generalization.

[236] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin et al. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

[237] A. Jacovi, A. Caciularu, O. Goldman and Y. Goldberg. 2023. Stop Uploading Test Data in Plain Text: Practical Strategies for Mitigating Data Contamination by Evaluation Benchmarks. ArXiv:2305.10160 [cs].

[238] N. Jain, K. Saifullah, Y. Wen, J. Kirchenbauer, M. Shu, A. Saha, M. Goldblum, J. Geiping et al. 2023. Bring your own data! self-supervised evaluation for large language models. *arXiv preprint arXiv:23062.13651*.

[239] J. Jang, S. Kim, S. Ye, D. Kim, L. Logeswaran, M. Lee, K. Lee and M. Seo. 2023. Exploring the Benefits of Training Expert Language Models over Instruction Tuning. ArXiv:2302.03202 [cs].

[240] J. R. Jeliazkov, D. del Alamo and J. D. Karpiak. 2023. Esmfold hallucinates native-like protein sequences. *bioRxiv*.

[241] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang et al. 2023. Survey of Hallucination in Natural Language Generation. *ACM Computing Surveys*, 55(12):1–38.

[242] G. Jiang, M. Xu, S.-C. Zhu, W. Han, C. Zhang and Y. Zhu. 2022. Mpi: Evaluating and inducing personality in pre-trained language models. *arXiv preprint arXiv:2206.07550*.

[243] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang and Q. Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

[244] Z. Jin, J. Liu, Z. Lyu, S. Poff, M. Sachan, R. Mihalcea, M. Diab and B. Schölkopf. 2023. Can large language models infer causation from correlation?

[245] A. Jinich, S. Z. Nazia, A. V. Tellez, D. Rappoport, M. AlQuraishi and K. Rhee. 2022. Predicting enzyme substrate chemical structure with protein language models. *bioRxiv*, pages 2022–09.

[246] Jonathan Frankle [@jefrankle]. 2022. Louder for the people in the back: LARGE MODELS (GPT, DALLE) = DATABASES PROMPTS = QUERIES OUTPUTS = RESPONSES NNs find new relations w/in data. Anyone, no matter the resources, can study better querying langs and possibly beat a big model they could never afford to train.

[247] D. Jones. 2022. Development and evaluation of speech recognition for the Welsh language. In *Proceedings of the 4th Celtic Language Technology Workshop within LREC2022*, pages 52–59, Marseille, France. European Language Resources Association.

[248] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates et al. 2021. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

[249] J. Kaddour. 2022. Stop wasting my time! saving days of imagenet and bert training with latest weight averaging. *arXiv preprint arXiv:2209.14981*.

[250] J. Kaddour. 2023. The MiniPile Challenge for Data-Efficient Language Models. ArXiv:2304.08442 [cs].

[251] J. Kaddour, O. Key, P. Nawrot, P. Minervini and M. J. Kusner. 2023. No Train No Gain: Revisiting Efficient Training Algorithms For Transformer-based Language Models. ArXiv:2307.06440 [cs].

[252] J. Kaddour, L. Liu, R. Silva and M. Kusner. 2022. When do flat minima optimizers work? In *Advances in Neural Information Processing Systems*.

[253] J. Kaddour, A. Lynch, Q. Liu, M. J. Kusner and R. Silva. 2022. Causal machine learning: A survey and open problems. *arXiv preprint arXiv:2206.15475*.

[254] J. Kaddour, Y. Zhu, Q. Liu, M. J. Kusner and R. Silva. 2021. Causal Effect Inference for Structured Treatments. In *Advances in Neural Information Processing Systems*, volume 34, pages 24841–24854. Curran Associates, Inc.

[255] M. Kale, A. Siddhant, R. Al-Rfou, L. Xue, N. Constant and M. Johnson. 2021. nmT5 - is parallel data still relevant for pre-training massively multilingual language models? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 683–691, Online. Association for Computational Linguistics.

[256] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford et al. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

[257] A. Karpathy. 2023. Tokenization issues (tweet).

[258] D. M. Katz, M. J. Bommarito, S. Gao and P. Arredondo. 2023. Gpt-4 passes the bar exam. *Available at SSRN 4389233*.

[259] A. Kazemnejad, I. Padhi, K. N. Ramamurthy, P. Das and S. Reddy. 2023. The impact of positional encoding on length generalization in transformers. *arXiv preprint arXiv:2305.19466*.

[260] Z. Kenton, T. Everitt, L. Weidinger, I. Gabriel, V. Mikulik and G. Irving. 2021. Alignment of language agents. *arXiv preprint arXiv:2103.14659*.

[261] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong and R. Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

[262] O. Khattab, K. Santhanam, X. L. Li, D. Hall, P. Liang, C. Potts and M. Zaharia. 2023. Demonstrate-Search-Predict: Composing retrieval and language models for knowledge-intensive NLP. ArXiv:2212.14024 [cs].

[263] D. Kiela, M. Bartolo, Y. Nie, D. Kaushik, A. Geiger, Z. Wu, B. Vidgen, G. Prasad et al. 2021. Dynabench: Rethinking benchmarking in nlp. *arXiv preprint arXiv:2104.14337*.

[264] J. Kim, M. Kim and B. Mozafari. 2022. Provable memorization capacity of transformers. In *The Eleventh International Conference on Learning Representations*.

[265] S. Kim, K. Mangalam, J. Malik, M. W. Mahoney, A. Gholami and K. Keutzer. 2023. Big little transformer decoder. *arXiv preprint arXiv:2302.07863*.

[266] T. Kim. 2022. Revisiting the practical effectiveness of constituency parse extraction from pre-trained language models. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5398–5408, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

[267] L. N. Kinch, R. D. Schaeffer, A. Kryshtafovych and N. V. Grishin. 2021. Target classification in the 14th round of the critical assessment of protein structure prediction (casp14). *Proteins: Structure, Function, and Bioinformatics*, 89(12):1618–1632.

[268] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers and T. Goldstein. 2023. A Watermark for Large Language Models. ArXiv:2301.10226 [cs].

[269] J. Kirchenbauer, J. Geiping, Y. Wen, M. Shu, K. Saifullah, K. Kong, K. Fernando, A. Saha et al. 2023. On the Reliability of Watermarks for Large Language Models. ArXiv:2306.04634 [cs].

[270] R. A. Klein, M. Vianello, F. Hasselman, B. G. Adams, R. B. Adams Jr, S. Alper, M. Aveyard, J. R. Axt et al. 2018. Many labs 2: Investigating variation in replicability across samples and settings. *Advances in Methods and Practices in Psychological Science*, 1(4):443–490.

[271] D. Kocetkov, R. Li, L. B. Allal, J. Li, C. Mou, C. M. Ferrandis, Y. Jernite, M. Mitchell et al. 2022. The stack: 3 tb of permissively licensed source code.

[272] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza et al. 2023. Chatgpt: Jack of all trades, master of none.

[273] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo and Y. Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*.

[274] A. Köpf, Y. Kilcher, D. von Rütte, S. Anagnostidis, Z.-R. Tam, K. Stevens, A. Barhoum, N. M. Duc et al. 2023. Openassistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*.

[275] T. Korbak, K. Shi, A. Chen, R. Bhalerao, C. L. Buckley, J. Phang, S. R. Bowman and E. Perez. 2023. Pretraining language models with human preferences. *arXiv preprint arXiv:2302.08582*.

[276] D. M. Korngiebel and S. D. Mooney. 2021. Considering the possibilities and pitfalls of generative pre-trained transformer 3 (gpt-3) in healthcare delivery. *NPJ Digital Medicine*, 4(1):1–3.

[277] M. Kosinski. 2023. Theory of mind may have spontaneously emerged in large language models.

[278] B. Krause, A. D. Gotmare, B. McCann, N. S. Keskar, S. Joty, R. Socher and N. F. Rajani. 2021. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[279] D. C. Krawczyk. 2018. Introduction to reasoning. *Reasoning—The Neuroscience of How We Think; Academic Press: Cambridge, MA, USA*, pages 1–11.

[280] K. Krishna, Y. Song, M. Karpinska, J. Wieting and M. Iyyer. 2023. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. ArXiv:2303.13408 [cs].

[281] T. Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia. Association for Computational Linguistics.

[282] T. Kudo and J. Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

[283] A. Kulkarni. 2021. GitHub Copilot AI Is Leaking Functional API Keys.

[284] S. R. Künzel, J. S. Sekhon, P. J. Bickel and B. Yu. 2019. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the national academy of sciences*, 116(10):4156–4165.

[285] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. Yu, J. Gonzalez, H. Zhang et al. 2023. vllm: Easy, fast, and cheap llm serving with pagedattention.

[286] E. Kıcıman, R. Ness, A. Sharma and C. Tan. 2023. Causal reasoning and large language models: Opening a new frontier for causality.

[287] P. Lab. 2023. Awesome-Prompt-Engineering. Original-date: 2023-02-09T18:22:52Z.

[288] A. K. Lampinen, S. C. Chan, I. Dasgupta, A. J. Nam and J. X. Wang. 2023. Passive learning of active causal strategies in agents and language models. *arXiv preprint arXiv:2305.16183*.

[289] H. Laurençon, L. Saulnier, T. Wang, C. Akiki, A. V. del Moral, T. L. Scao, L. V. Werra, C. Mou et al. 2022. The bigscience ROOTS corpus: A 1.6TB composite multilingual dataset. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[290] A. Lazaridou, E. Gribovskaya, W. Stokowiec and N. Grigorev. 2022. Internet-augmented language models through few-shot prompting for open-domain question answering.

[291] A. Lee, B. Miranda and S. Koyejo. 2023. Beyond Scale: the Diversity Coefficient as a Data Quality Metric Demonstrates LLMs are Pre-trained on Formally Diverse Data. ArXiv:2306.13840 [cs].

[292] D. Lee, J. Lee, J.-W. Ha, J.-H. Kim, S.-W. Lee, H. Lee and H. O. Song. 2023. Query-efficient black-box red teaming via bayesian optimization. *arXiv preprint arXiv:2305.17444*.

[293] K. Lee, O. Firat, A. Agarwal, C. Fannjiang and D. Sussillo. 2018. Hallucinations in neural machine translation.

[294] K. Lee, D. Ippolito, A. Nystrom, C. Zhang, D. Eck, C. Callison-Burch and N. Carlini. 2021. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*.

[295] N. Lee, W. Ping, P. Xu, M. Patwary, P. Fung, M. Shoeybi and B. Catanzaro. Factuality Enhanced Language Models for Open-Ended Text Generation.

[296] P. Lee, S. Bubeck and J. Petro. 2023. Benefits, limits, and risks of gpt-4 as an ai chatbot for medicine. *New England Journal of Medicine*, 388(13):1233–1239.

[297] E. Lehman, E. Hernandez, D. Mahajan, J. Wulff, M. J. Smith, Z. Ziegler, D. Nadler, P. Szolovits et al. 2023. Do we still need clinical language models?

[298] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer et al. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding.

[299] B. Lester, R. Al-Rfou and N. Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[300] Y. Leviathan, M. Kalman and Y. Matias. 2022. Fast inference from transformers via speculative decoding. *arXiv preprint arXiv:2211.17192*.

[301] D. M. Levine, R. Tuwani, B. Kompa, A. Varma, S. G. Finlayson, A. Mehrotra and A. Beam. 2023. The diagnostic and triage accuracy of the gpt-3 artificial intelligence model. *medRxiv*, pages 2023–01.

[302] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal and L. Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models.

[303] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

[304] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

[305] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil et al. 2022. Solving quantitative reasoning problems with language models.

[306] B. Z. Li, M. Nye and J. Andreas. 2021. Implicit representations of meaning in neural language models. *arXiv preprint arXiv:2106.00737*.

[307] C. Li, A. A. Awan, H. Tang, S. Rajbhandari and Y. He. 2021. 1-bit lamb: Communication efficient large-scale large-batch training with lamb's convergence speed. *arXiv preprint arXiv:2104.06069*.

[308] D. Li, R. Shao, A. Xie, Y. Sheng, L. Zheng, J. E. Gonzalez, I. Stoica, X. Ma et al. 2023. How long can open-source llms truly promise on context length?

[309] H. Li, D. Guo, W. Fan, M. Xu and Y. Song. 2023. Multi-step jailbreaking privacy attacks on chatgpt. *arXiv preprint arXiv:2304.05197*.

[310] R. Li, J. Su, C. Duan and S. Zheng. 2020. Linear attention mechanism: An efficient attention for semantic segmentation. *arXiv preprint arXiv:2007.14902*.

[311] X. L. Li and P. Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online. Association for Computational Linguistics.

[312] Y. Li, Z. Lin, S. Zhang, Q. Fu, B. Chen, J.-G. Lou and W. Chen. 2022. On the advance of making language models better reasoners.

[313] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

[314] Z. Li, C. You, S. Bhojanapalli, D. Li, A. S. Rawat, S. J. Reddi, K. Ye, F. Chern et al. 2023. The Lazy Neuron Phenomenon: On Emergence of Activation Sparsity in Transformers. ArXiv:2210.06313 [cs, stat].

[315] L. Lian, B. Li, A. Yala and T. Darrell. 2023. Llm-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models.

[316] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence and A. Zeng. 2023. Code as policies: Language model programs for embodied control.

[317] P. P. Liang, C. Wu, L.-P. Morency and R. Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576. PMLR.

[318] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan et al. 2022. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*.

[319] O. Lieber, O. Sharir, B. Lenz and Y. Shoham. 2021. Jurassic-1: Technical details and evaluation. *White Paper. AI21 Labs*, 1.

[320] V. Liévin, C. E. Hother and O. Winther. 2022. Can large language models reason about medical questions? *arXiv preprint arXiv:2207.08143*.

[321] C.-C. Lin, A. Jaech, X. Li, M. R. Gormley and J. Eisner. 2020. Limitations of autoregressive models and their alternatives. *arXiv preprint arXiv:2010.11939*.

[322] J. Lin, A. Yang, J. Bai, C. Zhou, L. Jiang, X. Jia, A. Wang, J. Zhang et al. 2021. M6-10t: A sharing-delinking paradigm for efficient multi-trillion parameter pretraining. *arXiv preprint arXiv:2110.03888*.

[323] S. Lin, J. Hilton and O. Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.

[324] X. V. Lin, T. Mihaylov, M. Artetxe, T. Wang, S. Chen, D. Simig, M. Ott, N. Goyal et al. 2022. Few-shot learning with multilingual generative language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[325] Y.-T. Lin and Y.-N. Chen. 2023. Llm-eval: Unified multi-dimensional automatic evaluation for open-domain conversations with large language models. *arXiv preprint arXiv:2305.13711*.

[326] Z. Lin, H. Akin, R. Rao, B. Hie, Z. Zhu, W. Lu, A. dos Santos Costa, M. Fazel-Zarandi et al. 2022. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *BioRxiv*.

[327] W. Ling, D. Yogatama, C. Dyer and P. Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

[328] B. Liu, J. T. Ash, S. Goel, A. Krishnamurthy and C. Zhang. 2023. Exposing Attention Glitches with Flip-Flop Language Modeling. ArXiv:2306.00946 [cs].

[329] F. Liu, J. M. Eisenschlos, F. Piccinno, S. Krichene, C. Pang, K. Lee, M. Joshi, W. Chen et al. 2022. Deplot: One-shot visual language reasoning by plot-to-table translation. *arXiv preprint arXiv:2212.10505*.

[330] H. Liu, C. Sferrazza and P. Abbeel. 2023. Languages are rewards: Hindsight finetuning using human feedback. *arXiv preprint arXiv:2302.02676*.

[331] H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal and C. A. Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

[332] H. Liu, S. M. Xie, Z. Li and T. Ma. 2022. Same pre-training loss, better downstream: Implicit bias matters for language models. *ArXiv*, abs/2210.14199.

[333] N. F. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni and P. Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. ArXiv:2307.03172 [cs].

[334] R. Liu, C. Jia, J. Wei, G. Xu and S. Vosoughi. 2022. Quantifying and alleviating political bias in language models. *Artificial Intelligence*, 304:103654.

[335] R. Liu and N. B. Shah. 2023. ReviewerGPT? An Exploratory Study on Using Large Language Models for Paper Reviewing. ArXiv:2306.00622 [cs].

[336] S. Liu and Z. Wang. 2023. Ten lessons we have learned in the new" sparseland": A short handbook for sparse neural network researchers. *arXiv preprint arXiv:2302.02596*.

[337] X. Liu, X. Yang, L. Ouyang, G. Guo, J. Su, R. Xi, K. Yuan and F. Yuan. 2022. Protein language model predicts mutation pathogenicity and clinical prognosis. *bioRxiv*, pages 2022–09.

[338] Z. Liu, A. Bahety and S. Song. 2023. Reflect: Summarizing robot experiences for failure explanation and correction.

[339] Z. Liu, E. Gan and M. Tegmark. 2023. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *arXiv preprint arXiv:2305.08746*.

[340] S. Longpre, L. Hou, T. Vu, A. Webson, H. W. Chung, Y. Tay, D. Zhou, Q. V. Le et al. 2023. The flan collection: Designing data and methods for effective instruction tuning.

[341] S. Longpre, G. Yauney, E. Reif, K. Lee, A. Roberts, B. Zoph, D. Zhou, J. Wei et al. 2023. A Pretrainer's Guide to Training Data: Measuring the Effects of Data Age, Domain Coverage, Quality, & Toxicity. ArXiv:2305.13169 [cs].

[342] Y. Lu, M. Bartolo, A. Moore, S. Riedel and P. Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

[343] Y. Lu, C. Li, M. Zhang, C. De Sa and Y. He. 2022. Maximizing communication efficiency for large-scale training via 0/1 adam. *arXiv preprint arXiv:2202.06009*.

[344] N. Lukas, A. Salem, R. Sim, S. Tople, L. Wutschitz and S. Zanella-Béguelin. 2023. Analyzing Leakage of Personally Identifiable Information in Language Models. ArXiv:2302.00539 [cs].

[345] B. Luo, R. Y. Lau, C. Li and Y.-W. Si. 2022. A critical review of state-of-the-art chatbot designs and applications. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 12(1):e1434.

[346] Y. Luo, N. Tang, G. Li, C. Chai, W. Li and X. Qin. 2021. Synthesizing natural language to visualization (nl2vis) benchmarks from nl2sql benchmarks. In *Proceedings of the 2021 International Conference on Management of Data*, pages 1235–1247.

[347] A. Lynch, G. J. Dovonon, J. Kaddour and R. Silva. 2023. Spawrious: A benchmark for fine control of spurious correlation biases. *arXiv preprint arXiv:2303.05470*.

[348] P. Ma, Z. Li, A. Sun and S. Wang. 2023. "oops, did i just say that?" testing and repairing unethical suggestions of large language models with suggest-critique-reflect process. *arXiv preprint arXiv:2305.02626*.

[349] X. Ma, G. Fang and X. Wang. 2023. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*.

[350] X. Ma, X. Kong, S. Wang, C. Zhou, J. May, H. Ma and L. Zettlemoyer. 2021. Luna: Linear unified nested attention. *Advances in Neural Information Processing Systems*, 34:2441–2453.

[351] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri et al. 2023. Self-refine: Iterative refinement with self-feedback.

[352] A. Madani, B. Krause, E. R. Greene, S. Subramanian, B. P. Mohr, J. M. Holton, J. L. Olmos Jr, C. Xiong et al. 2023. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, pages 1–8.

[353] M. Maddela, M. Ung, J. Xu, A. Madotto, H. Foran and Y.-L. Boureau. 2023. Training Models to Generate, Recognize, and Reframe Unhelpful Thoughts. ArXiv:2307.02768 [cs].

[354] S. Mahdavi, R. Liao and C. Thrampoulidis. 2023. Memorization Capacity of Multi-Head Attention in Transformers. ArXiv:2306.02010 [cs].

[355] S. Malladi, T. Gao, E. Nichani, A. Damian, J. D. Lee, D. Chen and S. Arora. 2023. Fine-Tuning Language Models with Just Forward Passes. ArXiv:2305.17333 [cs].

[356] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada and S. Paul. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

[357] P. Maniatis and D. Tarlow. 2023. Large sequence models for software development activities. Available from: https://ai.googleblog.com/2023/05/large-sequence-models-for-software.html. Accessed: 26/06/2023.

[358] R. R. McCrae and P. T. Costa Jr. 1997. Personality trait structure as a human universal. *American psychologist*, 52(5):509.

[359] I. R. McKenzie, A. Lyzhov, M. Pieler, A. Parrish, A. Mueller, A. Prabhu, E. McLean, A. Kirtland et al. 2023. Inverse Scaling: When Bigger Isn't Better. ArXiv:2306.09479 [cs].

[360] K. Meng, D. Bau, A. J. Andonian and Y. Belinkov. 2022. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*.

[361] K. Meng, A. S. Sharma, A. J. Andonian, Y. Belinkov and D. Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

[362] J. Menick, M. Trebacz, V. Mikulik, J. Aslanides, F. Song, M. Chadwick, M. Glaese, S. Young et al. 2022. Teaching language models to support answers with verified quotes.

[363] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick et al. 2023. Augmented language models: a survey. *arXiv preprint arXiv:2302.07842*.

[364] S. Milgram. 1963. Behavioral study of obedience. *The Journal of abnormal and social psychology*, 67(4):371.

[365] S. Min, K. Krishna, X. Lyu, M. Lewis, W.-t. Yih, P. W. Koh, M. Iyyer, L. Zettlemoyer et al. 2023. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. ArXiv:2305.14251 [cs].

[366] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis, H. Hajishirzi and L. Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work?

[367] M. Miotto, N. Rossberg and B. Kleinberg. 2022. Who is gpt-3? an exploration of personality, values and demographics. *arXiv preprint arXiv:2209.14338*.

[368] P. Mirowski, K. W. Mathewson, J. Pittman and R. Evans. 2022. Co-writing screenplays and theatre scripts with language models: An evaluation by industry professionals. *arXiv preprint arXiv:2209.14958*.

[369] A. Mishra, J. A. Latorre, J. Pool, D. Stosic, D. Stosic, G. Venkatesh, C. Yu and P. Micikevicius. 2021. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*.

[370] S. Mishra, D. Khashabi, C. Baral and H. Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland. Association for Computational Linguistics.

[371] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning and C. Finn. 2023. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. ArXiv:2301.11305 [cs].

[372] E. Mitchell, C. Lin, A. Bosselut, C. Finn and C. D. Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

[373] E. Mitchell, C. Lin, A. Bosselut, C. D. Manning and C. Finn. 2022. Memory-based model editing at scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 15817–15831. PMLR.

[374] R. Moriconi, M. P. Deisenroth and K. Sesh Kumar. 2020. High-dimensional bayesian optimization using low-dimensional feature spaces. *Machine Learning*, 109:1925–1943.

[375] M. Moussaïd, J. E. Kämmer, P. P. Analytis and H. Neth. 2013. Social influence and the collective dynamics of opinion formation. *PloS one*, 8(11):e78433.

[376] M. Mozes, J. Hoffmann, K. Tomanek, M. Kouate, N. Thain, A. Yuan, T. Bolukbasi and L. Dixon. 2023. Towards agile text classifiers for everyone. *arXiv preprint arXiv:2302.06541*.

[377] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T. L. Scao, M. S. Bari, S. Shen et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

[378] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi and A. Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707*.

[379] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

[380] N. Nanda, L. Chan, T. Lieberum, J. Smith and J. Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*.

[381] S. Nerella, S. Bandyopadhyay, J. Zhang, M. Contreras, S. Siegel, A. Bumin, B. Silva, J. Sena et al. 2023. Transformers in healthcare: A survey.

[382] A. Nguyen, N. Karampatziakis and W. Chen. 2023. Meet in the middle: A new pre-training paradigm. *arXiv preprint arXiv:2303.07295*.

[383] E. Nguyen, M. Poli, M. Faizi, A. Thomas, C. Birch-Sykes, M. Wornow, A. Patel, C. Rabideau et al. 2023. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *arXiv preprint arXiv:2306.15794*.

[384] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever and M. Chen. 2022. Glide: Towards photorealistic image generation and editing with text-guided diffusion models.

[385] X. Nie and S. Wager. 2021. Quasi-oracle estimation of heterogeneous treatment effects. *Biometrika*, 108(2):299–319.

[386] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese and C. Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis.

[387] F. Niu, B. Recht, C. Re, S. J. Wright and W. D. St. Hogwild!: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent.

[388] H. Nori, N. King, S. M. McKinney, D. Carignan and E. Horvitz. 2023. Capabilities of gpt-4 on medical challenge problems.

[389] K. Nottingham, P. Ammanabrolu, A. Suhr, Y. Choi, H. Hajishirzi, S. Singh and R. Fox. 2023. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling. *arXiv preprint arXiv:2301.12050*.

[390] S. Nurk, S. Koren, A. Rhie, M. Rautiainen, A. V. Bzikadze, A. Mikheenko, M. R. Vollger, N. Altemose et al. 2022. The complete sequence of a human genome. *Science*, 376(6588):44–53.

[391] M. Nye, A. J. Andreassen, G. Gur-Ari, H. Michalewski, J. Austin, D. Bieber, D. Dohan, A. Lewkowycz et al. 2021. Show your work: Scratchpads for intermediate computation with language models.

[392] Ofir Press [@OfirPress]. 2022. GPT-3 seems to be nondeterministic even when it should be (i.e. temperature == 0). Has anyone else noticed this? Is there a known fix? Video by my collaborator Muru Zhang. https://t.co/dOWYWPBYyP.

[393] N. Oh, G.-S. Choi and W. Y. Lee. 2023. Chatgpt goes to operating room: Evaluating gpt-4 performance and its potential in surgical education and training in the era of large language models. *medRxiv*.

[394] C. Olah. Mechanistic Interpretability, Variables, and the Importance of Interpretable Bases.

[395] C. Olsson, N. Elhage, N. Nanda, N. Joseph, N. DasSarma, T. Henighan, B. Mann, A. Askell et al. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

[396] OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/. Accessed: 2023-02-18.

[397] OpenAI. 2023. Chat gpt 4 painfully slow. https://community.openai.com/t/chat-gpt-4-painfully-slow/117996.

[398] OpenAI. 2023. Gpt-4 technical report.

[399] P. J. Ortiz Su'arez, B. Sagot and L. Romary. 2019. Asynchronous pipelines for processing huge corpora on medium to low resource infrastructures. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-7) 2019. Cardiff, 22nd July 2019*, pages 9 – 16, Mannheim. Leibniz-Institut f"ur Deutsche Sprache.

[400] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier and M. Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

[401] N. Ousidhoum, X. Zhao, T. Fang, Y. Song and D.-Y. Yeung. 2021. Probing toxic content in large pre-trained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4262–4274.

[402] C. Outeiral and C. Deane. 2022. Codon language embeddings provide strong signals for protein engineering. *bioRxiv*, pages 2022–12.

[403] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal et al. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

[404] M. Pagliardini, D. Paliotta, M. Jaggi and F. Fleuret. 2023. Faster causal attention over large sequences through sparse flash attention.

[405] J. Pan, T. Gao, H. Chen and D. Chen. 2023. What in-context learning "learns" in-context: Disentangling task recognition and task learning.

[406] B. Paranjape, S. Lundberg, S. Singh, H. Hajishirzi, L. Zettlemoyer and M. T. Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models.

[407] G. Park, B. Park, S. J. Kwon, B. Kim, Y. Lee and D. Lee. 2022. nuqmm: Quantized matmul for efficient inference of large-scale generative language models. *arXiv preprint arXiv:2206.09557*.

[408] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang and M. S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior.

[409] P. S. Park, P. Schoenegger and C. Zhu. 2023. Artificial intelligence in psychology research. *arXiv preprint arXiv:2302.07267*.

[410] A. Patel, B. Li, M. S. Rasooli, N. Constant, C. Raffel and C. Callison-Burch. 2023. Bidirectional language models are also few-shot learners.

[411] N. D. Patson, E. S. Darowski, N. Moon and F. Ferreira. 2009. Lingering misinterpretations in garden-path sentences: evidence from a paraphrasing task. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 35(1):280.

[412] D. Patterson, J. Gonzalez, U. Hölzle, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. R. So et al. 2022. The carbon footprint of machine learning training will plateau, then shrink. *Computer*, 55(7):18–28.

[413] A. Paullada, I. D. Raji, E. M. Bender, E. Denton and A. Hanna. 2021. Data and its (dis) contents: A survey of dataset development and use in machine learning research. *Patterns*, 2(11):100336.

[414] M. Pellert, C. M. Lechner, C. Wagner, B. Rammstedt and M. Strohmaier. 2023. Ai psychometrics: Using psychometric inventories to obtain psychological profiles of large language models.

[415] G. Penedo, Q. Malartic, D. Hesslow, R. Cojocaru, A. Cappelli, H. Alobeidli, B. Pannier, E. Almazrouei et al. 2023. The RefinedWeb Dataset for Falcon LLM: Outperforming Curated Corpora with Web Data, and Web Data Only. ArXiv:2306.01116 [cs].

[416] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, H. Cao, X. Cheng, M. Chung et al. 2023. RWKV: Reinventing RNNs for the Transformer Era. ArXiv:2305.13048 [cs].

[417] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, H. Cao, X. Cheng, M. Chung et al. 2023. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*.

[418] C. Peng, X. Yang, A. Chen, K. E. Smith, N. PourNejatian, A. B. Costa, C. Martin, M. G. Flores et al. 2023. A study of generative large language model for medical research and healthcare.

[419] Y. Peng. 2021. A MARVS analysis of two Chinese near-synonymous verbs of jumping based on Chinese corpora. In *Proceedings of the 35th Pacific Asia Conference on Language, Information and Computation*, pages 483–492, Shanghai, China. Association for Computational Linguistics.

[420] E. Perez, S. Huang, F. Song, T. Cai, R. Ring, J. Aslanides, A. Glaese, N. McAleese et al. 2022. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

[421] E. Perez, S. Ringer, K. Lukošiūtė, K. Nguyen, E. Chen, S. Heiner, C. Pettit, C. Olsson et al. 2022. Discovering language model behaviors with model-written evaluations.

[422] F. Perez and I. Ribeiro. 2022. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*.

[423] L. Peric, S. Mijic, D. Stammbach and E. Ash. 2020. Legal language modeling with transformers. In *Proceedings of the Fourth Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2020) held online in conjunction with te 33rd International Conference on Legal Knowledge and Information Systems (JURIX 2020) December 9, 2020*, volume 2764. CEUR-WS.

[424] B. Peters and A. F. T. Martins. 2021. Smoothing and shrinking the sparse Seq2Seq search space. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2642–2654, Online. Association for Computational Linguistics.

[425] J. Peters, D. Janzing and B. Schölkopf. 2017. *Elements of causal inference: foundations and learning algorithms*. The MIT Press.

[426] A. Petrov, E. La Malfa, P. H. Torr and A. Bibi. 2023. Language model tokenizers introduce unfairness between languages. *arXiv preprint arXiv:2305.15425*.

[427] T. Pettinato Oltz. 2023. Chatgpt, professor of law. *Professor of Law (February 4, 2023)*.

[428] J. Pfeiffer, A. Rücklé, C. Poth, A. Kamath, I. Vulić, S. Ruder, K. Cho and I. Gurevych. 2020. AdapterHub: A framework for adapting transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 46–54, Online. Association for Computational Linguistics.

[429] S. Pichai. 2023. An important next step on our ai journey. `https://blog.google/technology/ai/bard-google-ai-search-updates/`. Accessed: 2023-02-18.

[430] M. Poli, S. Massaroli, E. Nguyen, D. Y. Fu, T. Dao, S. Baccus, Y. Bengio, S. Ermon et al. 2023. Hyena Hierarchy: Towards Larger Convolutional Language Models. ArXiv:2302.10866 [cs].

[431] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, A. Levskaya, J. Heek, K. Xiao et al. 2022. Efficiently Scaling Transformer Inference. ArXiv:2211.05102 [cs].

[432] R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, A. Levskaya, J. Heek, K. Xiao et al. 2022. Efficiently scaling transformer inference. *arXiv preprint arXiv:2211.05102*.

[433] V. Prabhakaran, A. Mostafazadeh Davani and M. Diaz. 2021. On releasing annotator-level labels and information in datasets. In *Proceedings of the Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, pages 133–138, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[434] O. Press, N. A. Smith and M. Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation.

[435] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith and M. Lewis. 2023. Measuring and Narrowing the Compositionality Gap in Language Models. ArXiv:2210.03350 [cs].

[436] J. Qian, H. Wang, Z. Li, S. Li and X. Yan. 2022. Limitations of language models in arithmetic and symbolic induction. *arXiv preprint arXiv:2208.05051*.

[437] J. Rabelo, R. Goebel, M.-Y. Kim, Y. Kano, M. Yoshioka and K. Satoh. 2022. Overview and discussion of the competition on legal information Extraction/Entailment (COLIEE) 2021. *The Review of Socionetwork Strategies*, 16(1):111–133.

[438] A. Radford, R. Jozefowicz and I. Sutskever. 2017. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*.

[439] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey and I. Sutskever. 2022. Robust Speech Recognition via Large-Scale Weak Supervision. ArXiv:2212.04356 [cs, eess].

[440] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever. 2019. Language models are unsupervised multitask learners.

[441] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.

[442] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning and C. Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

[443] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li et al. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

[444] S. Rajbhandari, C. Li, Z. Yao, M. Zhang, R. Y. Aminabadi, A. A. Awan, J. Rasley and Y. He. 2022. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18332–18346. PMLR.

[445] S. Rajbhandari, J. Rasley, O. Ruwase and Y. He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press.

[446] S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith and Y. He. 2021. Zero-infinity: Breaking the gpu memory wall for extreme scale deep learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA. Association for Computing Machinery.

[447] I. D. Raji, E. M. Bender, A. Paullada, E. Denton and A. Hanna. 2021. Ai and the everything in the whole wide world benchmark. *arXiv preprint arXiv:2111.15366*.

[448] A. Rajkomar, E. Loreaux, Y. Liu, J. Kemp, B. Li, M.-J. Chen, Y. Zhang, A. Mohiuddin et al. 2022. Deciphering clinical abbreviations with a privacy protecting machine learning system. *Nature Communications*, 13(1):7456.

[449] R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi and Y. Choi. 2022. Is reinforcement learning (not) for natural language processing?: Benchmarks, baselines, and building blocks for natural language policy optimization. *arXiv preprint arXiv:2210.01241*.

[450] J. Rasley, S. Rajbhandari, O. Ruwase and Y. He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

[451] P. P. Ray. 2023. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, 3:121–154.

[452] E. Razumovskaia, J. Maynez, A. Louis, M. Lapata and S. Narayan. 2022. Little red riding hood goes around the globe: Crosslingual story planning and generation with large language models. *arXiv preprint arXiv:2212.10471*.

[453] B. Recht, C. Re, S. Wright and F. Niu. 2011. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24.

[454] J. Ren, S. Rajbhandari, R. Y. Aminabadi, O. Ruwase, S. Yang, M. Zhang, D. Li and Y. He. 2021. {ZeRO-Offload}: Democratizing {Billion-Scale} model training. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pages 551–564.

[455] X. Ren, P. Zhou, X. Meng, X. Huang, Y. Wang, W. Wang, P. Li, X. Zhang et al. 2023. Pangu-Sigma: Towards trillion parameter language model with sparse heterogeneous computing.

[456] Riley Goodside [@goodside]. 2022. An edge-case in GPT-3 with big implications: Inference is non-deterministic (even at temperature=0) when top-2 token probabilities are <1% different. So temperature=0 output is *very close* to deterministic, but actually isn't. Worth remembering.

[457] X. Robin, J. Haas, R. Gumienny, A. Smolinski, G. Tauriello and T. Schwede. 2021. Continuous automated model evaluation (cameo)—perspectives on the future of fully automated evaluation of structure prediction methods. *Proteins: Structure, Function, and Bioinformatics*, 89(12):1977–1986.

[458] A. Rohrbach, L. A. Hendricks, K. Burns, T. Darrell and K. Saenko. 2018. Object hallucination in image captioning. *arXiv preprint arXiv:1809.02156*.

[459] S. Roller, S. Sukhbaatar, A. Szlam and J. Weston. 2021. Hash layers for large sparse models.

[460] G. M. Rosa, L. Bonifacio, V. Jeronymo, H. Abonizio, R. Lotufo and R. Nogueira. 2022. Billions of parameters are worth more than in-domain training data: A case study in the legal case entailment task. *arXiv preprint arXiv:2205.15172*.

[461] L. Ross, D. Greene and P. House. 1977. The "false consensus effect": An egocentric bias in social perception and attribution processes. *Journal of experimental social psychology*, 13(3):279–301.

[462] Y. Rottenstreich and C. K. Hsee. 2001. Money, kisses, and electric shocks: On the affective psychology of risk. *Psychological science*, 12(3):185–190.

[463] A. Roush. You probably don't know how to do Prompt Engineering, let me educate you.

[464] L. Ruis, A. Khan, S. Biderman, S. Hooker, T. Rocktäschel and E. Grefenstette. 2022. Large language models are not zero-shot communicators.

[465] J. Rumbelow and mwatkins. SolidGoldMagikarp (plus, prompt generation).

[466] S. Russell. 2021. Human-compatible artificial intelligence. *Human-like machine intelligence*, pages 3–23.

[467] P. Rust, J. F. Lotz, E. Bugliarello, E. Salesky, M. de Lhoneux and D. Elliott. 2023. Language Modelling with Pixels. ArXiv:2207.06991 [cs].

[468] A. Sabne. 2020. Xla : Compiling machine learning for peak performance.

[469] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang and S. Feizi. 2023. Can AI-Generated Text be Reliably Detected? ArXiv:2303.11156 [cs].

[470] M. Safdari, G. Serapio-García, C. Crepy, S. Fitz, P. Romero, L. Sun, M. Abdulhai, A. Faust et al. 2023. Personality traits in large language models.

[471] S. Sagawa, P. W. Koh, T. B. Hashimoto and P. Liang. 2020. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization.

[472] O. Sainz, J. C. Campos, I. García-Ferrero, J. Etxaniz and E. Agirre. lm-contamination.

[473] L. Salewski, S. Alaniz, I. Rio-Torto, E. Schulz and Z. Akata. 2023. In-context impersonation reveals large language models' strengths and biases. *arXiv preprint arXiv:2305.14930*.

[474] G. Sanchez, H. Fan, A. Spangher, E. Levi, P. S. Ammanamanchi and S. Biderman. 2023. Stay on topic with Classifier-Free Guidance. ArXiv:2306.17806 [cs].

[475] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler et al. 2022. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations*.

[476] S. Sanyal, J. Kaddour, A. Kumar and S. Sanghavi. 2023. Understanding the effectiveness of early weight averaging for training large language models.

[477] E. Saravia. 2022. Prompt Engineering Guide. Publication Title: https://github.com/dair-ai/Prompt-Engineering-Guide original-date: 2022-12-16T16:04:50Z.

[478] J. Savelka, K. D. Ashley, M. A. Gray, H. Westermann and H. Xu. 2023. Explaining legal concepts with augmented large language models (gpt-4).

[479] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni et al. 2022. Bloom: A 176b-parameter open-access multilingual language model.

[480] R. Schaeffer, B. Miranda and S. Koyejo. 2023. Are emergent abilities of large language models a mirage?

[481] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda and T. Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

[482] T. Schick, J. Dwivedi-Yu, Z. Jiang, F. Petroni, P. Lewis, G. Izacard, Q. You, C. Nalmpantis et al. 2022. Peer: A collaborative language model.

[483] T. Schick and H. Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352.

[484] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[485] M. Schuster and K. Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152.

[486] T. Schuster, R. Schuster, D. J. Shah and R. Barzilay. 2020. The limitations of stylometry for detecting machine-generated fake news. *Computational Linguistics*, 46(2):499–510.

[487] R. Schwartz, J. Dodge, N. A. Smith and O. Etzioni. 2019. Green AI. ArXiv:1907.10597 [cs, stat].

[488] S. H. Schwartz, B. Breyer and D. Danner. 2015. Human values scale (ess). *Zusammenstellung sozialwissenschaftlicher Items und Skalen (ZIS)*.

[489] A. See, A. Pappu, R. Saxena, A. Yerukola and C. D. Manning. 2019. Do massively pretrained language models make better storytellers? In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 843–861, Hong Kong, China. Association for Computational Linguistics.

[490] R. Sennrich, B. Haddow and A. Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

[491] E. Sezgin, J. Sirrianni, S. L. Linwood et al. 2022. Operationalizing and implementing pretrained, large artificial intelligence linguistic models in the us health care system: Outlook of generative pretrained transformer 3 (gpt-3) as a service model. *JMIR Medical Informatics*, 10(2):e32875.

[492] P. Shaw, J. Uszkoreit and A. Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

[493] N. Shazeer. 2019. Fast transformer decoding: One write-head is all you need.

[494] N. Shazeer. 2019. Fast transformer decoding: One write-head is all you need.

[495] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton and J. Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

[496] Z. Shen, M. Zhang, H. Zhao, S. Yi and H. Li. 2021. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 3531–3539.

[497] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, B. Chen, P. Liang, C. Ré et al. 2023. High-throughput generative inference of large language models with a single gpu.

[498] T. Shevlane, S. Farquhar, B. Garfinkel, M. Phuong, J. Whittlestone, J. Leung, D. Kokotajlo, N. Marchal et al. 2023. Model evaluation for extreme risks. *arXiv preprint arXiv:2305.15324*.

[499] A. Shirafuji, Y. Watanobe, T. Ito, M. Morishita, Y. Nakamura, Y. Oda and J. Suzuki. 2023. Exploring the robustness of large language models for solving programming problems.

[500] O. Shliazhko, A. Fenogenova, M. Tikhonova, V. Mikhailov, A. Kozlova and T. Shavrina. 2022. mgpt: Few-shot learners go multilingual. *arXiv preprint arXiv:2204.07580*.

[501] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper and B. Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

[502] K. Shridhar, J. Macina, M. El-Assady, T. Sinha, M. Kapur and M. Sachan. 2022. Automatic generation of socratic subquestions for teaching math word problems. *ArXiv*, abs/2211.12835.

[503] K. Shridhar, A. Stolfo and M. Sachan. 2022. Distilling multi-step reasoning capabilities of large language models into smaller models via semantic decompositions. *arXiv preprint arXiv:2212.00193*.

[504] D. Shrivastava, H. Larochelle and D. Tarlow. 2022. Repository-level prompt generation for large language models of code. *arXiv preprint arXiv:2206.12839*.

[505] R. W. Shuai, J. A. Ruffolo and J. J. Gray. 2021. Generative language modeling for antibody design. *bioRxiv*, pages 2021–12.

[506] I. Shumailov, Z. Shumaylov, Y. Zhao, Y. Gal, N. Papernot and R. Anderson. 2023. The curse of recursion: Training on generated data makes models forget.

[507] K. Shuster, S. Poff, M. Chen, D. Kiela and J. Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

[508] K. Shuster, J. Xu, M. Komeili, D. Ju, E. M. Smith, S. Roller, M. Ung, M. Chen et al. 2022. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage.

[509] S. Sia and K. Duh. 2023. In-context learning as maintaining coherency: A study of on-the-fly machine translation using large language models. *ArXiv*, abs/2305.03573.

[510] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason et al. 2022. Progprompt: Generating situated robot task plans using large language models.

[511] K. Singhal, S. Azizi, T. Tu, S. S. Mahdavi, J. Wei, H. W. Chung, N. Scales, A. Tanwani et al. 2022. Large language models encode clinical knowledge.

[512] K. Singhal, T. Tu, J. Gottweis, R. Sayres, E. Wulczyn, L. Hou, K. Clark, S. Pfohl et al. 2023. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

[513] A. Sinitsin, D. Pyrkin, A. Babenko, V. Plokhotnyuk and S. Popov. 2020. EDITABLE NEURAL NETWORKS.

[514] S. L. Smith, P.-J. Kindermans, C. Ying and Q. V. Le. 2017. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*.

[515] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

[516] I. Solaiman and C. Dennison. 2021. Process for adapting language models to society (palms) with values-targeted datasets. *Advances in Neural Information Processing Systems*, 34:5861–5873.

[517] S. Soltan, S. Ananthakrishnan, J. FitzGerald, R. Gupta, W. Hamza, H. Khan, C. Peris, S. Rawls et al. 2022. Alexatm 20b: Few-shot learning using a large-scale multilingual seq2seq model. *arXiv preprint arXiv:2208.01448*.

[518] B. Sorscher, R. Geirhos, S. Shekhar, S. Ganguli and A. S. Morcos. 2022. Beyond neural scaling laws: beating power law scaling via data pruning. *arXiv preprint arXiv:2206.14486*.

[519] A. Srivastava, A. Rastogi, A. Rao, A. A. M. Shoeb, A. Abid, A. Fisch, A. R. Brown, A. Santoro et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

[520] J. Steinhardt. 2022. Future ml systems will be qualitatively different. *Accessed May*, 20:2022.

[521] J. Steinhardt. 2023. Emergent deception and emergent optimization. Available from: https://bounded-regret.ghost.io/emergent-deception-optimization/. Accessed: 29/04/2023.

[522] M. Stern, N. Shazeer and J. Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 10107–10116, Red Hook, NY, USA. Curran Associates Inc.

[523] C. Stevenson, I. Smal, M. Baas, R. Grasman and H. van der Maas. 2022. Putting gpt-3's creativity to the (alternative uses) test. *arXiv preprint arXiv:2206.08932*.

[524] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei et al. 2020. Learning to summarize with human feedback. In *Conference on Neural Information Processing Systems*.

[525] A. Stolfo, Z. Jin, K. Shridhar, B. Schölkopf and M. Sachan. 2022. A causal framework to quantify the robustness of mathematical reasoning with language models.

[526] J. Su, Y. Lu, S. Pan, B. Wen and Y. Liu. 2021. Roformer: Enhanced transformer with rotary position embedding.

[527] M. Sun, Z. Liu, A. Bair and J. Z. Kolter. 2023. A simple and effective pruning approach for large language models.

[528] T. Sun, Y. Shao, H. Qian, X. Huang and X. Qiu. 2022. Black-box tuning for language-model-as-a-service. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20841–20855. PMLR.

[529] X. Sun, T. Ge, F. Wei and H. Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5937–5947, Online. Association for Computational Linguistics.

[530] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

[531] Z. Sun. 2023. A short survey of viewing large language models in legal aspect.

[532] D. Surís, S. Menon and C. Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*.

[533] Susan Zhang [@suchenzang]. 2023. Piling on to the pile-on (sorry - it's always easy to criticize), here's a rant about benchmarks for LLMs that are used to back claims of "stronger" or "better" models. Let's start with a tour through GPT-3's Appendix G... 1/8.

[534] M. Suzgun, N. Scales, N. Schärli, S. Gehrmann, Y. Tay, H. W. Chung, A. Chowdhery, Q. V. Le et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

[535] S. Swaminathan, A. Dedieu, R. V. Raju, M. Shanahan, M. Lazaro-Gredilla and D. George. 2023. Schema-learning and rebinding as mechanisms of in-context learning and emergence. ArXiv:2307.01201 [cs].

[536] H. Tang, S. Gan, A. A. Awan, S. Rajbhandari, C. Li, X. Lian, J. Liu, C. Zhang et al. 2021. 1-bit adam: Communication efficient large-scale training with adam's convergence speed. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10118–10129. PMLR.

[537] L. Tang, G. Uberti and T. Shlomi. 2023. Baselines for Identifying Watermarked Large Language Models. ArXiv:2305.18456 [cs].

[538] L. Tang, Z. Sun, B. Idnay, J. G. Nestor, A. Soroush, P. A. Elias, Z. Xu, Y. Ding et al. 2023. Evaluating large language models on medical evidence summarization. *medRxiv*, pages 2023–04.

[539] R. Tang, Y.-N. Chuang and X. Hu. 2023. The Science of Detecting LLM-Generated Texts. ArXiv:2303.07205 [cs].

[540] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang and T. B. Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model.

[541] Y. Tay, D. Bahri, D. Metzler, D.-C. Juan, Z. Zhao and C. Zheng. 2021. Synthesizer: Rethinking self-attention for transformer models. In *International conference on machine learning*, pages 10183–10192. PMLR.

[542] Y. Tay, M. Dehghani, S. Abnar, H. W. Chung, W. Fedus, J. Rao, S. Narang, V. Q. Tran et al. 2022. Scaling laws vs model architectures: How does inductive bias influence scaling?

[543] Y. Tay, M. Dehghani, D. Bahri and D. Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

[544] Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama et al. 2022. Scale Efficiently: Insights from Pre-training and Fine-tuning Transformers. ArXiv:2109.10686 [cs].

[545] Y. Tay, M. Dehghani, V. Q. Tran, X. Garcia, J. Wei, X. Wang, H. W. Chung, D. Bahri et al. 2022. Ul2: Unifying language learning paradigms.

[546] Y. Tay, V. Q. Tran, S. Ruder, J. Gupta, H. W. Chung, D. Bahri, Z. Qin, S. Baumgartner et al. 2022. Charformer: Fast character transformers via gradient-based subword tokenization.

[547] Y. Tay, J. Wei, H. W. Chung, V. Q. Tran, D. R. So, S. Shakeri, X. Garcia, H. S. Zheng et al. 2022. Transcending scaling laws with 0.1% extra compute.

[548] R. Taylor, M. Kardas, G. Cucurull, T. Scialom, A. Hartshorn, E. Saravia, A. Poulton, V. Kerkez et al. 2022. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*.

[549] W. L. Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

[550] J. Thiergart, S. Huber and T. Übellacker. 2021. Understanding emails and drafting responses–an approach using gpt-3. *arXiv preprint arXiv:2102.03062*.

[551] R. Thoppilan, D. De Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

[552] R. Tian, S. Narayan, T. Sellam and A. P. Parikh. 2020. Sticking to the Facts: Confident Decoding for Faithful Data-to-Text Generation. ArXiv:1910.08684 [cs].

[553] K. Tirumala, A. H. Markosyan, L. Zettlemoyer and A. Aghajanyan. Memorization Without Overfitting: Analyzing the Training Dynamics of Large Language Models.

[554] H. Q. To, N. D. Bui, J. Guo and T. N. Nguyen. 2023. Better language models of code through self-improvement. *arXiv preprint arXiv:2304.01228*.

[555] A. Tornede, D. Deng, T. Eimer, J. Giovanelli, A. Mohan, T. Ruhkopf, S. Segel, D. Theodorakopoulos et al. 2023. AutoML in the Age of Large Language Models: Current Challenges, Future Opportunities and Risks. ArXiv:2306.08107 [cs].

[556] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal et al. 2023. LLaMA: Open and Efficient Foundation Language Models. ArXiv:2302.13971 [cs].

[557] H. Touvron, L. Martin and K. Stone. Llama 2: Open Foundation and Fine-Tuned Chat Models.

[558] C. Tran, S. Khadkikar and A. Porollo. 2023. Survey of protein sequence embedding models. *International Journal of Molecular Sciences*, 24(4):3775.

[559] A. Uchendu, T. Le, K. Shu and D. Lee. 2020. Authorship Attribution for Neural Text Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8384–8395, Online. Association for Computational Linguistics.

[560] J. Uesato, N. Kushman, R. Kumar, F. Song, N. Siegel, L. Wang, A. Creswell, G. Irving et al. 2022. Solving math word problems with process- and outcome-based feedback.

[561] S. University. 2023. Holistic evaluation of langauge models results page. Available from: https://crfm.stanford.edu/helm/latest/?groups=1. Accessed: 23/03/2023.

[562] K. Valmeekam, A. Olmo, S. Sreedharan and S. Kambhampati. 2023. Large language models still can't plan (a benchmark for llms on planning and reasoning about change).

[563] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser and I. Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[564] S. Vemprala, R. Bonatti, A. Bucker and A. Kapoor. 2023. Chatgpt for robotics: Design principles and model abilities.

[565] A. Venigalla, J. Frankle and M. Carbin. 2022. Pubmed gpt: A domain- specific large language model for biomedical text. https://www.mosaicml.com/blog/introducing-pubmed-gpt. Accessed: 2023-01-24.

[566] R. Verkuil, O. Kabeli, Y. Du, B. I. Wicky, L. F. Milles, J. Dauparas, D. Baker, S. Ovchinnikov et al. 2022. Language models generalize beyond natural proteins. *bioRxiv*, pages 2022–12.

[567] A. Vijayakumar, M. Cogswell, R. Selvaraju, Q. Sun, S. Lee, D. Crandall and D. Batra. 2018. Diverse beam search for improved description of complex scenes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

[568] P. Villalobos, J. Sevilla, L. Heim, T. Besiroglu, M. Hobbhahn and A. Ho. 2022. Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv preprint arXiv:2211.04325*.

[569] H. Viswanath and T. Zhang. 2023. Fairpy: A toolkit for evaluation of social biases and their mitigation in large language models. *arXiv preprint arXiv:2302.05508*.

[570] J. von Oswald, E. Niklasson, E. Randazzo, J. Sacramento, A. Mordvintsev, A. Zhmoginov and M. Vladymyrov. 2022. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*.

[571] H. d. Vries. 2023. Go smol or go home.

[572] T. Vu, B. Lester, N. Constant, R. Al-Rfou' and D. Cer. 2022. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland. Association for Computational Linguistics.

[573] J. P. Wahle, T. Ruas, T. Foltýnek, N. Meuschke and B. Gipp. 2022. Identifying machine-paraphrased plagiarism. In *International Conference on Information*, pages 393–413. Springer.

[574] J. P. Wahle, T. Ruas, F. Kirstein and B. Gipp. 2022. How large language models are transforming machine-paraphrased plagiarism. *arXiv preprint arXiv:2210.03568*.

[575] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy and S. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

[576] B. Wang and A. Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

[577] C. Wang, K. Cho and J. Gu. 2020. Neural machine translation with byte-level subwords. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9154–9160.

[578] C. Wang, X. Liu, Z. Chen, H. Hong, J. Tang and D. Song. 2022. DeepStruct: Pretraining of language models for structure prediction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 803–823, Dublin, Ireland. Association for Computational Linguistics.

[579] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan and A. Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*.

[580] H. Wang, J. Kaddour, S. Liu, J. Tang, M. Kusner, J. Lasenby and Q. Liu. 2022. Evaluating self-supervised learning for molecular graph embeddings. *arXiv preprint arXiv:2206.08005*.

[581] P. Wang, L. Li, L. Chen, D. Zhu, B. Lin, Y. Cao, Q. Liu, T. Liu et al. 2023. Large Language Models are not Fair Evaluators. ArXiv:2305.17926 [cs].

[582] R. Wang, H. Wang, F. Mi, Y. Chen, R. Xu and K.-F. Wong. 2023. Self-critique prompting with large language models for inductive instructions. *arXiv preprint arXiv:2305.13733*.

[583] S. Wang, Y. Liu, Y. Xu, C. Zhu and M. Zeng. 2021. Want to reduce labeling cost? gpt-3 can help.

[584] S. Wang, S. Menon, T. Long, K. Henderson, D. Li, K. Crowston, M. Hansen, J. V. Nickerson et al. 2023. Reelframer: Co-creating news reels on social media with generative ai. *arXiv preprint arXiv:2304.09653*.

[585] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery and D. Zhou. 2022. Self-consistency improves chain of thought reasoning in language models.

[586] Y. Wang, Z. Yu, Z. Zeng, L. Yang, C. Wang, H. Chen, C. Jiang, R. Xie et al. 2023. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*.

[587] Y. Wang. 2021. Comment section personalization: Algorithmic, interface, and interaction design. In *Proceedings of the EACL Hackashop on News Media Content Analysis and Automated Report Generation*, pages 84–88, Online. Association for Computational Linguistics.

[588] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi and H. Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions.

[589] Y. Wang, S. Mishra, P. Alipoormolabashi, Y. Kordi, A. Mirzaei, A. Naik, A. Ashok, A. S. Dhanasekaran et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109.

[590] Y. Wang, Y. Zhao and L. Petzold. 2023. Are large language models ready for healthcare? a comparative study on clinical language understanding.

[591] Z. Wang, S. Cai, A. Liu, X. Ma and Y. Liang. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*.

[592] Z. Wang, J. Wohlwend and T. Lei. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

[593] Z. Wang, Z. Dai, B. Póczos and J. Carbonell. 2019. Characterizing and avoiding negative transfer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11293–11302.

[594] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, N. De Freitas et al. 2013. Bayesian optimization in high dimensions via random embeddings. In *IJCAI*, volume 13, pages 1778–1784.

[595] T. Webb, K. J. Holyoak and H. Lu. 2022. Emergent analogical reasoning in large language models.

[596] A. Webson and E. Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States. Association for Computational Linguistics.

[597] A. Wei, N. Haghtalab and J. Steinhardt. 2023. Jailbroken: How Does LLM Safety Training Fail? ArXiv:2307.02483 [cs].

[598] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai et al. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

[599] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma et al. 2022. Emergent abilities of large language models.

[600] J. Wei, Y. Tay and Q. V. Le. 2022. Inverse scaling can become u-shaped. *arXiv preprint arXiv:2211.02011*.

[601] J. Wei, X. Wang, D. Schuurmans, M. Bosma, brian ichter, F. Xia, E. H. Chi, Q. V. Le et al. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

[602] L. Weidinger, J. Mellor, M. Rauh, C. Griffin, J. Uesato, P.-S. Huang, M. Cheng, M. Glaese et al. 2021. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*.

[603] M. Weiss. 2019. Deepfake bot submissions to federal public comment websites cannot be distinguished from human submissions. *Technology Science*, 2019121801.

[604] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho and J. Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

[605] L. Weng. 2023. Large transformer model inference optimization. *Lil'Log*.

[606] L. Weng. 2023. Prompt engineering. *lilianweng.github.io*.

[607] M. Willig, M. ZEČEVIĆ, D. S. Dhami and K. Kersting. 2023. Causal parrots: Large language models may talk causality but are not causal. *preprint*.

[608] F. Winkelmolen, N. Ivkin, H. F. Bozkurt and Z. Karnin. 2020. Practical and sample efficient zero-shot hpo. *arXiv preprint arXiv:2007.13382*.

[609] Y. Wolf, N. Wies, Y. Levine and A. Shashua. 2023. Fundamental limitations of alignment in large language models. *arXiv preprint arXiv:2304.11082*.

[610] M. Wornow, Y. Xu, R. Thapa, B. Patel, E. Steinberg, S. Fleming, M. A. Pfeffer, J. Fries et al. 2023. The shaky foundations of clinical foundation models: A survey of large language models and foundation models for emrs.

[611] F. Wu, D. Radev and J. Xu. 2023. When geometric deep learning meets pretrained protein language models. *bioRxiv*, pages 2023–01.

[612] J. Wu, L. Ouyang, D. M. Ziegler, N. Stiennon, R. Lowe, J. Leike and P. Christiano. 2021. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*.

[613] J. Wu, F. Wu, B. Jiang, W. Liu and P. Zhao. 2022. tfold-ab: Fast and accurate antibody structure prediction without sequence homologs. *bioRxiv*, pages 2022–11.

[614] P. Y. Wu, J. A. Tucker, J. Nagler and S. Messing. 2023. Large language models can be used to estimate the ideologies of politicians in a zero-shot learning setting.

[615] S. Wu, X. Zhao, T. Yu, R. Zhang, C. Shen, H. Liu, F. Li, H. Zhu et al. 2021. Yuan 1.0: Large-scale pre-trained language model in zero-shot and few-shot learning.

[616] S. Wu, O. Irsoy, S. Lu, V. Dabravolski, M. Dredze, S. Gehrmann, P. Kambadur, D. Rosenberg et al. 2023. Bloomberggpt: A large language model for finance.

[617] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

[618] Y. Wu, M. Gardner, P. Stenetorp and P. Dasigi. 2022. Generating data to mitigate spurious correlations in natural language inference datasets. *arXiv preprint arXiv:2203.12942*.

[619] Z. Wu, L. Qiu, A. Ross, E. Akyürek, B. Chen, B. Wang, N. Kim, J. Andreas et al. 2023. Reasoning or Reciting? Exploring the Capabilities and Limitations of Language Models Through Counterfactual Tasks. ArXiv:2307.02477 [cs].

[620] Y. Xiao and W. Y. Wang. 2021. On Hallucination and Predictive Uncertainty in Conditional Language Generation. ArXiv:2103.15025 [cs].

[621] Q. Xie, Z. Luo, B. Wang and S. Ananiadou. 2023. A survey on biomedical text summarization with pre-trained language model.

[622] S. M. Xie, H. Pham, X. Dong, N. Du, H. Liu, Y. Lu, P. Liang, Q. V. Le et al. 2023. DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining. ArXiv:2305.10429 [cs].

[623] S. M. Xie, A. Raghunathan, P. Liang and T. Ma. 2022. An Explanation of In-context Learning as Implicit Bayesian Inference. ArXiv:2111.02080 [cs].

[624] S. M. Xie, S. Santurkar, T. Ma and P. Liang. 2023. Data Selection for Language Models via Importance Resampling. ArXiv:2302.03169 [cs].

[625] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao and D. Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

[626] F. F. Xu, U. Alon, G. Neubig and V. J. Hellendoorn. 2022. A systematic evaluation of large language models of code.

[627] M. Xu, X. Yuan, S. Miret and J. Tang. 2023. Protst: Multi-modality learning of protein sequences and biomedical texts. *arXiv preprint arXiv:2301.12040*.

[628] Y. Xu, H. Lee, D. Chen, B. Hechtman, Y. Huang, R. Joshi, M. Krikun, D. Lepikhin et al. 2021. Gspmd: general and scalable parallelization for ml computation graphs. *arXiv preprint arXiv:2105.04663*.

[629] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts and C. Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. ArXiv:2105.13626 [cs].

[630] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts and C. Raffel. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

[631] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua and C. Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

[632] L. Yan, L. Sha, L. Zhao, Y. Li, R. Martinez-Maldonado, G. Chen, X. Li, Y. Jin et al. 2023. Practical and ethical challenges of large language models in education: A systematic literature review.

[633] G. Yang, E. Hu, I. Babuschkin, S. Sidor, X. Liu, D. Farhi, N. Ryder, J. Pachocki et al. 2021. Tuning large neural networks via zero-shot hyperparameter transfer. *Advances in Neural Information Processing Systems*, 34:17084–17097.

[634] J. Yang, H. Jin, R. Tang, X. Han, Q. Feng, H. Jiang, B. Yin and X. Hu. 2023. Harnessing the power of llms in practice: A survey on chatgpt and beyond.

[635] K. Yang and D. Klein. 2021. Fudge: Controlled text generation with future discriminators. *arXiv preprint arXiv:2104.05218*.

[636] K. Yang, D. Klein, N. Peng and Y. Tian. 2022. Doc: Improving long story coherence with detailed outline control. *arXiv preprint arXiv:2212.10077*.

[637] K. Yang, N. Peng, Y. Tian and D. Klein. 2022. Re3: Generating longer stories with recursive reprompting and revision. *arXiv preprint arXiv:2210.06774*.

[638] X. Yang, K. Chen, W. Zhang, C. Liu, Y. Qi, J. Zhang, H. Fang and N. Yu. 2023. Watermarking Text Generated by Black-Box Language Models. ArXiv:2305.08883 [cs].

[639] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao and K. Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. ArXiv:2305.10601 [cs].

[640] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan and Y. Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

[641] X. Yao, Y. Zheng, X. Yang and Z. Yang. 2022. NLP From Scratch Without Large-Scale Pretraining: A Simple and Efficient Framework. In *Proceedings of the 39th International Conference on Machine Learning*, pages 25438–25451. PMLR. ISSN: 2640-3498.

[642] Y. Yao, P. Wang, B. Tian, S. Cheng, Z. Li, S. Deng, H. Chen and N. Zhang. 2023. Editing Large Language Models: Problems, Methods, and Opportunities. ArXiv:2305.13172 [cs].

[643] Z. Yao, R. Y. Aminabadi, M. Zhang, X. Wu, C. Li and Y. He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *arXiv preprint arXiv:2206.01861*.

[644] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C. D. Manning, P. Liang and J. Leskovec. 2022. Deep bidirectional language-knowledge graph pretraining. *arXiv preprint arXiv:2210.09338*.

[645] S. Yi, R. Goel, C. Khatri, A. Cervone, T. Chung, B. Hedayatnia, A. Venkatesh, R. Gabriel et al. 2019. Towards coherent and engaging spoken dialog response generation using automatic conversation evaluators. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 65–75, Tokyo, Japan. Association for Computational Linguistics.

[646] D. Yogatama, C. de Masson d'Autume and L. Kong. 2021. Adaptive semiparametric language models. *Transactions of the Association for Computational Linguistics*, 9:362–373.

[647] T. Yoneda, J. Fang, P. Li, H. Zhang, T. Jiang, S. Lin, B. Picker, D. Yunis et al. 2023. Statler: State-maintaining language models for embodied reasoning.

[648] K. M. Yoo, D. Park, J. Kang, S.-W. Lee and W. Park. 2021. GPT3Mix: Leveraging large-scale language models for text augmentation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2225–2239, Punta Cana, Dominican Republic. Association for Computational Linguistics.

[649] K. Yoo, W. Ahn, J. Jang and N. Kwak. 2023. Robust Natural Language Watermarking through Invariant Features. ArXiv:2305.01904 [cs].

[650] R. You, Y. Liu, H. Mamitsuka and S. Zhu. 2021. Bertmesh: deep contextual representation learning for large-scale high-performance mesh indexing with full text. *Bioinformatics*, 37(5):684–692.

[651] F. Yu, L. Quartey and F. Schilder. 2022. Legal prompting: Teaching a language model to think like a lawyer. *arXiv preprint arXiv:2212.01326*.

[652] L. Yu, D. Simig, C. Flaherty, A. Aghajanyan, L. Zettlemoyer and M. Lewis. 2023. Megabyte: Predicting million-byte sequences with multiscale transformers. *arXiv preprint arXiv:2305.07185*.

[653] P. Yu, M. Artetxe, M. Ott, S. Shleifer, H. Gong, V. Stoyanov and X. Li. 2022. Efficient language modeling with sparse all-mlp.

[654] P. Yu, T. Wang, O. Golovneva, B. Alkhamissy, G. Ghosh, M. Diab and A. Celikyilmaz. 2022. Alert: Adapting language models to reasoning tasks. *arXiv preprint arXiv:2212.08286*.

[655] L. Yunxiang, L. Zihan, Z. Kai, D. Ruilong and Z. You. 2023. Chatdoctor: A medical chat model fine-tuned on llama model using medical domain knowledge.

[656] E. Zelikman, Y. Wu, J. Mu and N. Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

[657] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner and Y. Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

[658] A. Zeng, X. Liu, Z. Du, Z. Wang, H. Lai, M. Ding, Z. Yang, Y. Xu et al. 2022. Glm-130b: An open bilingual pre-trained model.

[659] W. Zeng, X. Ren, T. Su, H. Wang, Y. Liao, Z. Wang, X. Jiang, Z. Yang et al. 2021. Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation.

[660] F. Zhang, B. Chen, Y. Zhang, J. Liu, D. Zan, Y. Mao, J.-G. Lou and W. Chen. 2023. Repocoder: Repository-level code completion through iterative retrieval and generation.

[661] H. Zhang, L. H. Li, T. Meng, K.-W. Chang and G. V. d. Broeck. 2022. On the Paradox of Learning to Reason from Data. ArXiv:2205.11502 [cs].

[662] H. Zhang, D. Duckworth, D. Ippolito and A. Neelakantan. 2021. Trading off diversity and quality in natural language generation. In *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*, pages 25–33, Online. Association for Computational Linguistics.

[663] M. Zhang and Y. He. 2020. Accelerating training of transformer-based language models with progressive layer dropping.

[664] M. Zhang, O. Press, W. Merrill, A. Liu and N. A. Smith. 2023. How Language Model Hallucinations Can Snowball. ArXiv:2305.13534 [cs].

[665] S. Zhang. 2023. [...] that's an unhelpful order of magnitude difference in how large of a model you should be training in order to be considered "compute optimal". https://twitter.com/suchenzang/status/1616752494608007171?s=20. Accessed: 2023-06-06.

[666] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab et al. 2022. Opt: Open pre-trained transformer language models.

[667] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

[668] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. McKeown and T. B. Hashimoto. 2023. Benchmarking large language models for news summarization.

[669] Z. Zhang, Y. Gu, X. Han, S. Chen, C. Xiao, Z. Sun, Y. Yao, F. Qi et al. 2021. Cpm-2: Large-scale cost-effective pre-trained language models.

[670] Z. Zhang, Y. Lin, Z. Liu, P. Li, M. Sun and J. Zhou. 2022. Moefication: Transformer feed-forward layers are mixtures of experts.

[671] Z. Zhang, A. Zhang, M. Li and A. Smola. 2022. Automatic chain of thought prompting in large language models.

[672] S. Zhao, J. Wen, L. A. Tuan, J. Zhao and J. Fu. 2023. Prompt as triggers for backdoor attack: Examining the vulnerability in language models. *arXiv preprint arXiv:2305.01219*.

[673] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang et al. 2023. A Survey of Large Language Models. ArXiv:2303.18223 [cs].

[674] Y. Zhao, A. Gu, R. Varma, L. Luo, C.-C. Huang, M. Xu, L. Wright, H. Shojanazeri et al. 2023. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*.

[675] Z. Zhao, E. Wallace, S. Feng, D. Klein and S. Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

[676] B. Zheng, L. Dong, S. Huang, S. Singhal, W. Che, T. Liu, X. Song and F. Wei. 2021. Allocating large vocabulary capacity for cross-lingual language model pre-training. *arXiv preprint arXiv:2109.07306*.

[677] L. Zheng, Z. Li, H. Zhang, Y. Zhuang, Z. Chen, Y. Huang, Y. Wang, Y. Xu et al. 2022. Alpa: Automating inter- and Intra-Operator parallelism for distributed deep learning. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 559–578, Carlsbad, CA. USENIX Association.

[678] R. Zheng, S. Dou, S. Gao, W. Shen, B. Wang, Y. Liu, S. Jin, Q. Liu et al. 2023. Secrets of RLHF in Large Language Models Part I: PPO. ArXiv:2307.04964 [cs].

[679] W. Zhong, R. Cui, Y. Guo, Y. Liang, S. Lu, Y. Wang, A. Saied, W. Chen et al. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

[680] A. Zhou, Y. Ma, J. Zhu, J. Liu, Z. Zhang, K. Yuan, W. Sun and H. Li. 2021. Learning N: M fine-grained structured sparse neural networks from scratch. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

[681] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat et al. 2023. LIMA: Less Is More for Alignment. ArXiv:2305.11206 [cs].

[682] D. Zhou, N. Schärli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, C. Cui et al. 2022. Least-to-most prompting enables complex reasoning in large language models.

[683] Y. Zhou, A. I. Muresanu, Z. Han, K. Paster, S. Pitis, H. Chan and J. Ba. 2023. Large language models are human-level prompt engineers. In *International Conference on Learning Representations*.

[684] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba and S. Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.

[685] B. Zhuang, J. Liu, Z. Pan, H. He, Y. Weng and C. Shen. 2023. A survey on efficient training of transformers. *arXiv preprint arXiv:2302.01107*.

[686] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano and G. Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

[687] B. Zoph, I. Bello, S. Kumar, N. Du, Y. Huang, J. Dean, N. Shazeer and W. Fedus. 2022. St-moe: Designing stable and transferable sparse expert models.

[688] M. Zvyagin, A. Brace, K. Hippe, Y. Deng, B. Zhang, C. O. Bohorquez, A. Clyde, B. Kale et al. 2022. Genslms: Genome-scale language models reveal sars-cov-2 evolutionary dynamics. *bioRxiv*, pages 2022–10.