PROMPT2MODEL: Generating Deployable Models from Natural Language Instructions

Vijay Viswanathan[*1], **Chenyang Zhao**[*2†], **Amanda Bertsch**[1], **Tongshuang Wu**[1], **Graham Neubig**[1]
[1]Carnegie Mellon University,
[2]BNRIST / Department of Computer Science and Technology, Tsinghua University

Abstract

Large language models (LLMs) enable system builders today to create competent NLP systems through prompting, where they only need to describe the task in natural language and provide a few examples. However, in other ways, LLMs are a step backward from traditional special-purpose NLP models; they require extensive computational resources for deployment and can be gated behind APIs. In this paper, we propose Prompt2Model, a general-purpose method that takes a natural language task description like the prompts provided to LLMs, and uses it to train a special-purpose model that is conducive to deployment. This is done through a multi-step process of retrieval of existing datasets and pretrained models, dataset generation using LLMs, and supervised fine-tuning on these retrieved and generated datasets. Over three tasks, we demonstrate that given the same few-shot prompt as input, Prompt2Model trains models that outperform the results of a strong LLM, `gpt-3.5-turbo`, by an average of 20% while being up to 700 times smaller. We also show that this data can be used to obtain reliable *performance estimates* of model performance, enabling model developers to assess model reliability before deployment. Prompt2Model is available open-source at `https://github.com/neulab/prompt2model`.[1]

1 Introduction

Traditionally, building an NLP model from scratch has been a substantial undertaking. An NLP practitioner seeking to solve a new problem would need to define their task scope, find or create data that specifies the intended system behavior, choose a suitable model architecture, train the model, assess its performance through evaluation, and then deploy it for real-world usage (Paleyes et al., 2022).

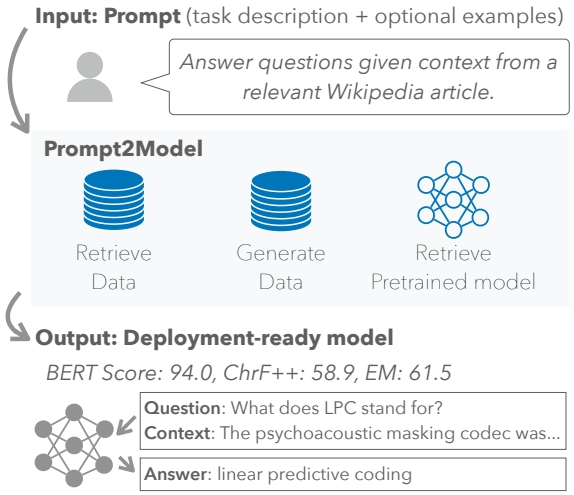

Figure 1: Prompt2Model is a framework for generating a small yet accurate model from a prompt.

LLMs like GPT-3 (Brown et al., 2020; Liu et al., 2023b) offer a lighter-weight paradigm for NLP system construction through "prompting" (Reynolds and McDonell, 2021). Practitioners can now write a prompt specifying the intended system behavior (optionally with a few demonstrations), and ask an LLM to generate a desired output via text completion. This makes it possible to prototype NLP systems rapidly for a variety of applications without writing a single line of code (Floridi and Chiriatti, 2020).

However, there is still a gap between proof-of-concept prototyping — showing LLMs can be prompted for a particular task — and practical deployment. Prompting LLMs can be expensive as they require either a significant amount of computing or access to commercial APIs, and their reliance on the input prompt quality makes them unstable compared to trained models (Min et al., 2022; Bubeck et al., 2023). Because practitioners usually lack annotated data, it is also more challenging for them to evaluate or debug their systems before deployment (Jiang et al., 2022). Additionally, practitioners have expressed concerns about

[*]equal contribution.
[†]Work done during an internship at Carnegie Mellon.
[1]Our demo video is posted at `youtu.be/LYYQ_EhGd-Q`.

Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 413–421
December 6-10, 2023

the high serving cost and slow prediction time associated with using LLMs (Park et al., 2022), and those working in high-stakes domains cannot rely on commercial LLM APIs due to privacy concerns.

In this work, we present Prompt2Model, a system that retains the ability to specify system behavior in a light-weight way through *prompting*, while still resulting in a *deployable special-purpose model*, maintaining all the advantages thereof. Prompt2Model is designed as an automated pipeline that extracts essential task information from users' prompts and then automatically collects and synthesizes task-specific knowledge through three channels:

- *Dataset retrieval*: Whenever possible, we retrieve task-relevant annotated data (Färber and Leisinger, 2021; Viswanathan et al., 2023).
- *Dataset generation*: We distill knowledge from an LLM ("teacher model") by employing it to generate a pseudo-labeled dataset (Wang et al., 2021; He et al., 2023; Gudibande et al., 2023).
- *Model retrieval*: Based on the prompt, we identify a pretrained language model appropriate for the user's intent. This model is fine-tuned and evaluated using the generated and retrieved data.

Prompt2Model is designed to support different instantiations of each of these components. We provide a reference implementation using a `gpt-3.5-turbo`-based dataset generator, a dataset retriever based on DataFinder (Viswanathan et al., 2023), and a BM25-based model retriever. We evaluate three tasks covering both traditional NLP benchmarks and novel applications and find that, empirically, Prompt2Model sometimes produces small models that outperform `gpt-3.5-turbo` using the same prompt. On 2 of these 3 tasks, we observe >20 point improvements over `gpt-3.5-turbo`, despite our finetuned model being up to 700 times smaller. We also find that Prompt2Model can generate effective evaluation datasets. Prompt2Model can serve the following purposes for the community:

1. **A tool for quickly building small and competent NLP systems**: Prompt2Model can produce task-specific models that outperform LLMs in a few hours without any manual data annotation or architecture design. The method bridges the gap between proof-of-concept LLM prototyping and practical deployment.
2. **A testbed for end-to-end, prompt-based model training**: Given Prompt2Model's extensible design, it can offer a platform for exploring new techniques in model distillation, dataset generation, synthetic evaluation, dataset retrieval, and model retrieval. Our platform allows studying these components using extrinsic downstream metrics, enabling empirical progress in these research areas.

2 Prompt2Model Framework

Prompt2Model provides a framework to automate machine learning pipelines: data collection, model training, evaluation, and deployment. We illustrate our automated pipeline in Figure 2. At the core is our automatic data collection system, leveraging dataset retrieval and LLM-based dataset generation to obtain labeled data. We then retrieve pretrained models and finetune them on the training datasets. Finally, we evaluate trained models on the test datasets and create a web UI for interaction.

Our general-purpose method is designed to be modular and extensible; each component can be implemented differently or disabled by a practitioner. We give an overview of our framework, then in section 3 we describe our reference implementation.

Prompt Parser As the primary input to our system, users provide prompts similar to those used for LLMs. These prompts comprise an instruction and, optionally, a few demonstrations of the anticipated behavior. While this open-ended interface is convenient for users, end-to-end ML pipelines may benefit from a *Prompt Parser* that processes this input, such as segmenting the prompt into an instruction and individual demonstrations.

Dataset Retriever Given a prompt, we first try to discover existing manually annotated data that support users' task descriptions. There are several design decisions for the *Dataset Retriever*:

1. What datasets to search against?
2. How to index datasets for search?
3. Which dataset columns are needed for the user's task, and which columns should be ignored?

We use *DataFinder* (Viswanathan et al., 2023) in our implementation, described in §3.2.

Dataset Generator Not all conceivable tasks have any existing relevant annotated data. To support a wider range of tasks, the *Dataset Generator* synthesizes training data from user-specific requirements parsed by the *Prompt Parser*. This presents challenges related to cost efficiency, speed, diver-

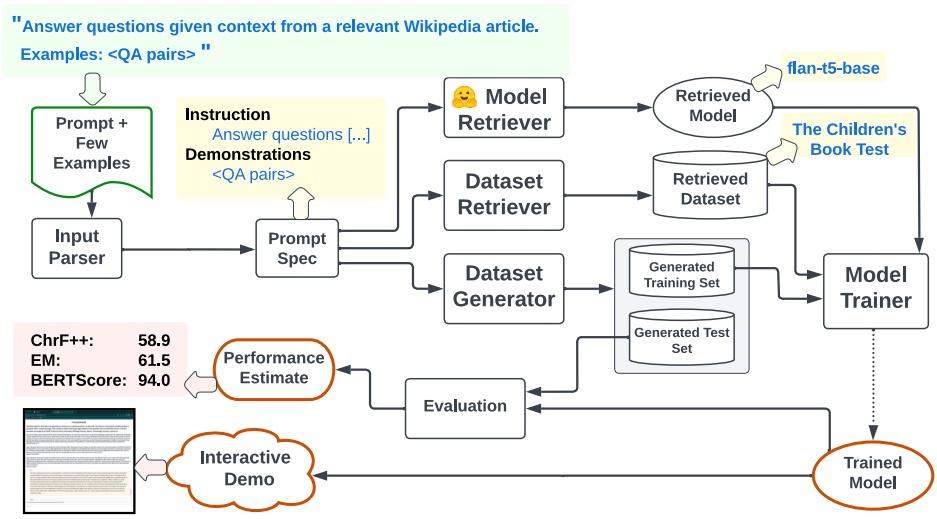

Figure 2: The Prompt2Model architecture seeks to automate the core machine learning development pipeline.

sity, and quality control. Our suggested solution to these challenges is described in §3.3.

Model Retriever Besides training data, we must identify an appropriate model to finetune. We cast this as a retrieval problem against model descriptions and metadata such as popularity or tasks supported. The reference implementation of our *Model Retriever*, described in §3.4, searches against models on Hugging Face (Wolf et al., 2020).

Training Given retrieved and generated datasets and a pretrained model, we use a *Model Trainer* to finetune the model on a subset of the data.

Evaluation After training models on a portion of the retrieved and generated datasets, we give the remaining data to an *Model Evaluator* module. Selecting the correct metrics for an arbitrary task is a difficult problem. We describe our suggested strategies for task-agnostic evaluation in §3.6.

2.1 Web App Creation

We include a component called the *Demo Creator* to create a user interface to interact with the model. We briefly describe our implementation in §3.7.

3 Reference Implementation

Prompt2Model is designed modularly to support customization of each component in our framework (described in §2), but we have provided a reference implementation to enable immediate adoption.

3.1 Prompt Parser

We parse the prompt into `instruction` and `demonstrations` fields (shown in Figure 2), where the instruction represents the task specification and the demonstrations exemplify the desired behavior. We utilize an LLM (OpenAI's `gpt-3.5-turbo-0613` in our experiments) to segment user prompts.

3.2 Dataset Retriever

To retrieve datasets for a prompt, we adapt the *DataFinder* system introduced by Viswanathan et al. (2023). We adapted *DataFinder* to user-generated dataset descriptions from Hugging Face Datasets (Lhoest et al., 2021). Once a relevant dataset is identified, the next step is to determine which columns of the dataset correspond to the input and the output specified by the user. As automatically inducing the correct schema for any dataset can be challenging, we adopt a human-in-the-loop approach. We present the top-k datasets, to the user and allow them to either select the most relevant dataset or to state that none are a good fit for their task. We then ask the user to identify the appropriate input and output columns.

3.3 Dataset Generator

We engineered our dataset generator to enable speed-optimized generation at a low cost while creating diverse and high-quality examples. Our strategy comprises the following components:

High-Diversity Few-Shot Prompting We use automated prompt engineering to generate diverse samples. We expand the user-provided demonstrations with a sample of previously generated examples to promote diversity and avoid duplicates.

Temperature Annealing We adjust the sampling temperature from low (favoring determinism) to high (encouraging exploration) proportional to the number of examples already generated to maintain output quality while gradually enabling diversity.

Minimum Bayes-Risk Decoding Given that LLM may generate multiple outputs for the same inputs, we use *self-consistency*, a form of Minimum Bayes Risk Decoding (Wang et al., 2022; Bertsch et al., 2023) to select pseudo-labels. Specifically, we create a consensus output for each unique input by selecting the most frequent answer; in the case of ties, we heuristically select the shortest answer.

3.4 Model Retriever

To select an appropriate model to fine-tune, we presently limit ourselves to encoder-decoder models on Hugging Face (Wolf et al., 2020) following work showing that encoder-decoder models are more data-efficient for model distillation (Calderon et al., 2023). This restriction still leaves a large set of pretrained models to choose from. Using the user's instruction as a query, we search against textual descriptions of models on Hugging Face.

This search task is challenging because Hugging Face model descriptions are sparse and contain lots of templatic text, often with only a few words that signify the content of the model. To address this, we follow the HyDE framework (Gao et al., 2023) and use `gpt-3.5-turbo` to create a *hypothetical model description* given the user's instructions (with an example shown in Figure 3). Using this as an expanded query, we use BM25 to compute query-model similarity scores (Robertson et al., 1995). For practical purposes, we filter out models whose size exceeds a user-specified threshold (set to 3GB by default). Using the intuition that highly-downloaded models are likely to be high in quality, we finally rank models by:

$$BM25(\text{query}, \text{model}) \cdot \log(\#\text{ of Downloads} + 1).$$

3.5 Training

Dataset Processing We train the model with up to two datasets- one generated and one retrieved. We treat all datasets as "text-to-text" (Raffel et al., 2020) by textualizing the input columns of each and prepending the user's instructions to the input.

Finetuning We concatenate the retrieved and generated datasets and shuffle them before fine-tuning. We use the same default hyperparameters for all tasks.[2] We train with the AdamW optimizer with `lr = 5e-5` for 3 epochs, which takes roughly one hour for all tasks.

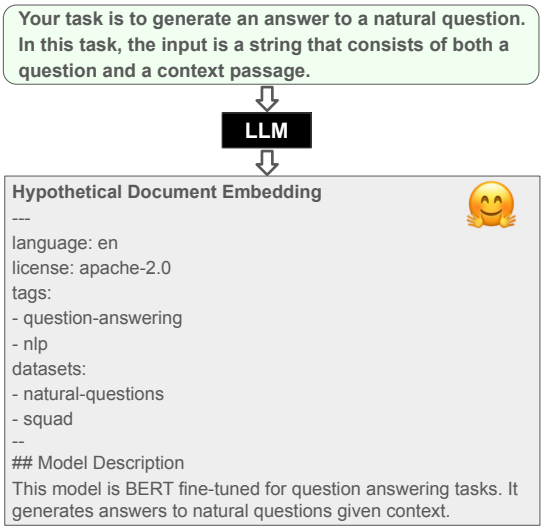

Figure 3: We expand queries to our model retriever by constructing a *hypothetical model description*.

3.6 Evaluation

We automatically evaluate models using three general-purpose metrics: `Exact Match`, `ChrF++` (Popović, 2015), and `BERTScore` (Zhang et al., 2019). `ChrF++` measures character and word overlap. `BERTScore` captures semantic similarity using embeddings of model outputs and references — we use XLM-R (Conneau et al., 2020) to compute embeddings to support multilingual evaluation.

3.7 Web App Creation

We automatically create a graphical user interface to interact with the trained model. This web application, built using Gradio (Abid et al., 2019), can be easily deployed publicly.

4 Experimental Setup

Tasks As a proof-of-concept, we test our system's ability to learn a model for three tasks:

- *Machine Reading Question Answering*: We use SQuAD (Rajpurkar et al., 2016) as ground truth to evaluate the setting where pretrained models and training datasets are plentiful.
- *Japanese NL-to-Code*: Code generation from Japanese-language queries is a challenging scenario where prior work exists but no annotated

[2]We empirically find that these default hyperparameters are effective, but we plan on implementing hyperparameter selection using generated validation data in the future.

Method	SQuAD (EM)	MCoNaLa (ChrF++)	Temporal (ChrF++)
Prompt2Model	61.5	13.1	55.2
w/o Model Ret.	61.5	15.8	55.2
w/o Data Ret.	50.2	16.6	N/A
`gpt-3.5-turbo`	42.1	37.3	30.7

Table 1: We evaluate the model produced by Prompt2Model on real benchmarks for each test set, compared to `gpt-3.5-turbo`, also used in our dataset generator. We also examine the effect of removing specific components — model retrieval and dataset retrieval. There are no relevant datasets available for the Temporal task, so we did not use retrieved data there.

data or pretrained models are available. We use MCoNaLa (Wang et al., 2023) for evaluation.

- *Temporal Expression Normalization*: We finally consider a task where no relevant models or data are available. We use the Temporal dataset of Wu et al. (2023) as ground truth for evaluation.

Though Prompt2Model offers automated model evaluation (on generated and retrieved datasets), we use real benchmark datasets here to measure Prompt2Model's ability to train accurate models.

LLM Baseline A primary goal of our work is to train small models that can match or outperform LLMs. To measure success towards this goal, we report the performance of `gpt-3.5-turbo` on each benchmark. We provide `gpt-3.5-turbo`[3] the same instruction and demonstrations provided to Prompt2Model, for fair comparison.

5 Experiment Results

5.1 Downstream performance

How effective is Prompt2Model at producing a high-quality model? In Table 1, we evaluated models produced by Prompt2Model and baseline LLM `gpt-3.5-turbo`, on real benchmark datasets for each task — SQuAD, MCoNaLa, and Temporal. We further examine the effect of removing 2 specific elements of the Prompt2Model pipeline — model retrieval and dataset retrieval.

On 2 of 3 datasets, we find that Prompt2Model produces models that are considerably more accurate than `gpt-3.5-turbo`. This is remarkable because the retrieved model for SQuAD and Temporal is Flan-T5, which has only 250M parameters.

We observe that Prompt2Model's performance on MCoNaLa's Japanese-to-Python task is significantly worse than `gpt-3.5-turbo`. One explanation is the relatively low diversity of the generated Japanese queries; 45 of 5000 examples are different ways of saying "find the maximum value in a list of numbers", suggesting that `gpt-3.5-turbo` may struggle to generate diverse text for non-English languages. Another reason is the lack of an appropriate student model — the retrieved models were trained on either multiple languages or codes, but not both.

5.2 Combining retrieved and generated datasets is powerful

Using SQuAD as a case study, we can compare Prompt2Model with a model trained on the same amount of data from the true dataset.[4] Our prompt for Prompt2Model is a description of the SQuAD passage-level question-answering task (Figure 1), and we exclude SQuAD from the retrieved datasets. We evaluate models finetuned on:

1. 3k examples from the closest retrieved dataset[5]
2. 3k examples generated by Prompt2Model
3. 1. + 2. (i.e. the full Prompt2Model pipeline)
4. 3k examples from SQuAD (analogous to the user custom-annotating data for a task).

Table 2 shows the results across these settings. While using only retrieved or generated data causes a reduction in performance, combining these two methods provides similar performance to using a subset of the original SQuAD. Compared to custom-annotating a subset of SQuAD for this task, Prompt2Model allows for *similar performance at less than 1% of the cost*.

5.3 Our generated evaluation data can identify real modeling improvements

High-quality generated data should *discriminate* between multiple candidate models to select a model that will perform well downstream. We finetune various models on a generated dataset and rank their performance on generated test data and the test data from the target (real) dataset. We evaluate Kendall's rank correlation (Kendall, 1938) between the two rankings to determine if our generated data is effective for model selection. This

[3] We used `gpt-3.5-turbo-0613`, accessed between July 26 and August 6, 2023.

[4] We focus on only SQuAD here because our other two datasets have small or nonexistent training splits.

[5] The closest dataset retrieved by the dataset retriever for our SQuAD-inspired prompt is The Children's Book Test Dataset (Hill et al., 2016).

Method	#Train	Performance	Anno. Cost
Retrieval only	3,000	56.79	≈ $ 0
Generation only	3,000	44.20	≈ $ 5
Retrieval+generation	6,000	61.46	≈ $ 5
Custom annotation	3,000	61.64	≈ $ 540

Table 2: We compare performance on SQuAD on an annotation-cost basis, using datasets produced by different modules of Prompt2Model, along with manual annotation. We use exact match accuracy on the real SQuAD test set[6] to measure *the true task performance*. The cost of custom annotation is calculated using the reported annotator pay rate of $9/hour from Rajpurkar et al. (2016) and keeping 1,000 validation examples.

Dataset	Metric	τ	p-value
SQuAD	EM	64.3	0.03*
Temporal	ChrF++	24.2	0.31
MCoNaLa (JP)	ChrF++	70.9	0.00**

Table 3: We evaluate 10 different models on real test sets and their corresponding generated clones. We compute Kendall's Tau on the ranked lists of models and find statistically significant correlations for 2 of 3 datasets.

is closely related to the concept of concurrence between benchmarks (Liu et al., 2023a).

Table 3 shows Kendall's τ for each task, computed over a set of reasonable models.[7] The generated data shows a strong correlation to the true performance on two of the three datasets.

6 Discussion and Conclusion

We propose Prompt2Model, a framework that automatically constructs task-specific models using only natural language prompts. Our proof-of-concept experiments show that, despite using a similar simple interface like LLMs, Prompt2Model delivers small yet accurate models and its generated datasets can be used to estimate real-world performance. Prompt2Model's extensible and modular design makes it a platform for advancing model distillation, dataset generation, synthetic evaluation, dataset retrieval, and model retrieval.

We believe our Prompt2Model framework can inspire various novel research questions. These questions may include how much data should we generate for downstream model training and how diverse should it be? How do we effectively mix retrieved and generated data to achieve complementary strengths (e.g. using dataset generation to focus on inputs that the retrieved dataset fails to cover)? Since users may struggle to articulate their needs, future extensions should address human-in-the-loop correction by enabling either iterative editing of prompts or post-hoc fixing of data and models when the task metadata extraction and generated data do not align with user intentions. We invite the community to contribute novel implementations of various components in our framework.

[7]This set of models consisted of 5 T5-family models, 2 BART-family models, and 1-5 additional retrieved models from the *Model Retriever*, depending on the task.

Limitations

Our paper uses proprietary LLM APIs in our experiments, which is problematic as a scientific artifact (Rogers et al., 2023). Our software supports open-source LLMs to avoid reliance on proprietary APIs.

Another limitation of our work is the limited ability of Prompt2Model to support tasks that require processing languages other than English. While we have shown the limitations of our system at supporting code generation from Japanese natural language queries, our system is likely to struggle more with lower-resource languages. We use the unpublished `gpt-3.5-turbo` model for our Dataset Generator in our reference implementation. This model is believed to be similar to GPT-3 (Brown et al., 2020), which was trained on 93% English documents. Our use of this model may exacerbate existing disparities between high-resource languages and low-resource languages.

One potential limitation is that we have only tested our approach on 3 tasks, each with a single dataset and a single evaluation metric. We justify this decision because our focus is on providing an extensible software system rather than establishing state-of-the-art results on many datasets, but we believe that our results suggest broader applicability.

Acknowledgements

VV was supported by a fellowship from NEC Laboratories Europe. CZ was supported in part by the Deng Feng Fund from the Beijing National Research Center for Information Science and Technology. We are grateful to Alex Cabrera, Will Epperson, Nelson Liu, Arjun Ramani, Zirui Cheng, Zhiyuan Zeng, Tianci Xue, Yanchen Liu, Yi-Hsin Hung and Zhilin Yang for their guidance. We particularly appreciate Zirui Cheng's video production support for our demo.

Ethics Statement

Any system which makes powerful technology more accessible to the public has ethical implications. Widder et al. (2022) discuss ethical issues with open-source packages in relation to software libraries for deepfaking, including the possibility of enabling malicious actors to use technology that they would otherwise not have the technical skills to leverage. This is also a risk for an AutoML system such as Prompt2Model; however, we believe this risk is outweighed by the benefits of greater accessibility, especially given that a low barrier to entry for generating harmful data already exists in the form of prompted, web-interface models.

While Prompt2Model could, if given harmful inputs, generate toxic, offensive, or inaccurate synthetic data, this is no more of a risk with Prompt2Model than it is with the underlying prompted model (Bender et al., 2021); indeed, the use of models and supplementary datasets retrieved from Hugging Face may lessen the likelihood of a downstream model replicating harms from the prompted model's outputs, though more investigation is needed. Like all ML models, the models that Prompt2Model returns can make mistakes, and we aim to be transparent in our documentation about potential limitations of the system.

We hope that Prompt2Model will be broadly useful. Our work is motivated by a desire to increase the accessibility of NLP models to people who are not in the NLP community but would benefit from the community's innovations; particularly, to people who would use NLP models downstream but may not have the domain-specific knowledge to design their system. Prompt2Model may also prove useful for early NLP researchers by providing a starting point for intuitions about baselines for various tasks and enabling the discovery of similarities between a described task and existing work. We open-source Prompt2Model and welcome community contributions.

References

Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. 2019. Gradio: Hassle-free sharing and testing of ML models in the wild. *arXiv preprint arXiv:1906.02569*.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Amanda Bertsch, Alex Xie, Graham Neubig, and Matthew R. Gormley. 2023. It's MBR all the way down: Modern generation techniques through the lens of minimum bayes risk. In *EMNLP 2023 Big Picture Workshop*, Singapore.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Nitay Calderon, Subhabrata Mukherjee, Roi Reichart, and Amir Kantor. 2023. A systematic study of knowledge distillation for natural language generation with pseudo-target training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14632–14659, Toronto, Canada. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 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.

Michael Färber and Ann-Kathrin Leisinger. 2021. Recommending datasets for scientific problem descriptions. In *CIKM*, pages 3014–3018.

Luciano Floridi and Massimo Chiriatti. 2020. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, 30:681–694.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Arnav Gudibande, Eric Wallace, Charles Burton Snell, Xinyang Geng, Hao Liu, P. Abbeel, Sergey Levine, and Dawn Song. 2023. The false promise of imitating proprietary llms. *ArXiv*, abs/2305.15717.

Xingwei He, Zheng-Wen Lin, Yeyun Gong, Alex Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2023. Annollm: Making large language models to be better crowdsourced annotators. *ArXiv*, abs/2303.16854.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations.

Ellen Jiang, Kristen Olson, Edwin Toh, Alejandra Molina, Aaron Donsbach, Michael Terry, and Carrie J Cai. 2022. Promptmaker: Prompt-based prototyping with large language models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–8.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nelson F. Liu, Tony Lee, Robin Jia, and Percy Liang. 2023a. Do question answering modeling improvements hold across benchmarks? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13186–13218, Toronto, Canada. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023b. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Andrei Paleyes, Raoul-Gabriel Urma, and Neil D Lawrence. 2022. Challenges in deploying machine learning: a survey of case studies. *ACM Computing Surveys*, 55(6):1–29.

Gunho Park, Baeseong Park, Minsub Kim, Sungjae Lee, Jeonghoon Kim, Beomseok Kwon, Se Jung Kwon, Byeongwook Kim, Youngjoo Lee, and Dongsoo Lee. 2022. Lut-gemm: Quantized matrix multiplication based on luts for efficient inference in large-scale generative language models.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI EA '21, New York, NY, USA. Association for Computing Machinery.

Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Mike Gatford, and A. Payne. 1995. Okapi at trec-4. In *Text Retrieval Conference*.

Anna Rogers, Niranjan Balasubramanian, Leon Derczynski, Jesse Dodge, Alexander Koller, Sasha Luccioni, Maarten Sap, Roy Schwartz, Noah A Smith, and Emma Strubell. 2023. Closed ai models make bad baselines.

Vijay Viswanathan, Luyu Gao, Tongshuang Wu, Pengfei Liu, and Graham Neubig. 2023. DataFinder: Scientific dataset recommendation from natural language descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10288–10303, Toronto, Canada. Association for Computational Linguistics.

Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, and Michael Zeng. 2021. Want to reduce labeling cost? GPT-3 can help. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4195–4205, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models.

Zhiruo Wang, Grace Cuenca, Shuyan Zhou, Frank F. Xu, and Graham Neubig. 2023. MCoNaLa: A benchmark for code generation from multiple natural languages. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 265–273, Dubrovnik, Croatia. Association for Computational Linguistics.

David Gray Widder, Dawn Nafus, Laura Dabbish, and James Herbsleb. 2022. Limits and possibilities for "ethical ai" in open source: A study of deepfakes. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 2035–2046, New York, NY, USA. Association for Computing Machinery.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Sherry Wu, Hua Shen, Daniel S Weld, Jeffrey Heer, and Marco Tulio Ribeiro. 2023. Scattershot: Interactive in-context example curation for text transformation. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, IUI '23, page 353–367, New York, NY, USA. Association for Computing Machinery.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert.