PROMPT2MODEL: Generating Deployable Models from Natural Language Instructions

Vijay Viswanathan[1*], **Chenyang Zhao**[1,2*],
Amanda Bertsch[1], **Tongshuang Wu**[1], **Graham Neubig**[1]
[1]Carnegie Mellon University, [2]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. 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 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).

Recently, the rise of LLMs like GPT-3 (Brown et al., 2020; Liu et al., 2023b) offers 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).

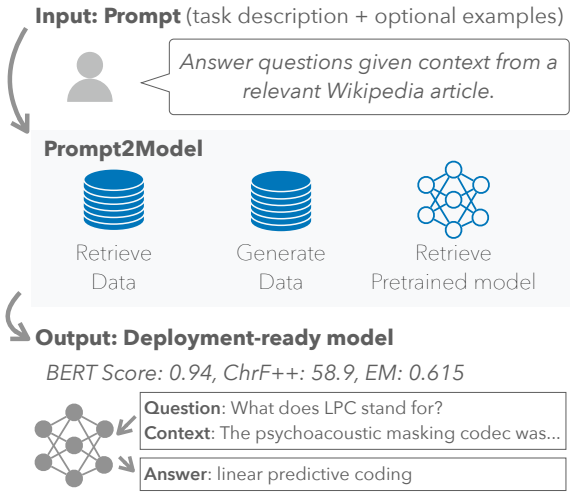

Figure 1: Prompt2Model is a framework for generating a small yet accurate model from a prompt.

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 compute 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 do not have enough annotated validation data to measure their system performance, it is also more challenging for them to *debug* their systems before deployment (Jiang et al., 2022). Additionally, LLM-prompted systems pose usability challenges. Practitioners have expressed concerns about the

*equal contribution.
[1]Our demo video is posted at `youtu.be/LYYQ_EhGd-Q`.

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. For instance, sharing user data with LLM service providers is illegal for many applications in the US (Sezgin et al., 2022).

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 collect training data by retrieving 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. Prior work has demonstrated that such a dataset can be used to train a smaller "student" model to emulate the behavior of the teacher model (Wang et al., 2021a; He et al., 2023; Gudibande et al., 2023).
- *Model retrieval*: Based on the prompt, we identify a pretrained language model whose parametric knowledge is appropriate for the user's intent. This chosen model serves as the student model and is further fine-tuned and evaluated using the generated and retrieved data.

Prompt2Model is designed to support different instantiations of each of these components. In experiments, we demonstrate its utility with a `gpt-3.5-turbo`-based dataset generator, a dataset retriever based on DataFinder (Viswanathan et al., 2023), and a model retriever using BM25. We evaluate on three tasks covering both traditional NLP benchmarks and novel applications, and find that, empirically, Prompt2Model is capable of producing small models that rival or outperform `gpt-3.5-turbo` when using the same prompt as input. On 2 of these 3 tasks, we observe >20 point improvements over the `gpt-3.5-turbo` baseline, despite the final model produced by Prompt2Model being up to 700 times smaller. We also find that we can generate effective evaluation datasets; performance improvements on these synthetic clones of real benchmark also hold on their real counterparts. We believe that Prompt2Model can serve the following purposes for the community:

1. **A tool for quickly building small and competent NLP systems**: Prompt2Model can be directly used to 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 the proof-of-concept LLM prototyping and the practical deployment of the model.
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 on these research areas.

2 Prompt2Model

Our system, Prompt2Model, provides a platform to automate the components of a machine learning pipeline: data collection, model training, evaluation, and deployment. We illustrate our automated pipeline in Figure 2. The core workhorse of our system is our automatic data collection system, which leverages dataset retrieval and LLM-based dataset generation to obtain labeled data relevant to the user's needs. We then retrieve pretrained models from Hugging Face (Wolf et al., 2020), which we finetune on the training splits of the collected datasets. Finally, we evaluate our trained models on the test splits of the same datasets and automatically create a web UI that can be used to interact with the model.

2.1 Prompt Parser

As the primary input to our system, users provide prompts similar to those utilized for LLMs. These prompts comprise an instruction and, optionally, a few demonstrations of the anticipated behavior. To support our pipeline, we parse the prompt into `instruction` and `demonstrations` fields (shown in Figure 2), where the instruction represents the primary task or objective and the demonstrations exemplify the desired behavior. For this purpose, we utilize an LLM with in-context learning to parse user prompts, employing the OpenAI `gpt-3.5-turbo-0613` in our experiments. If

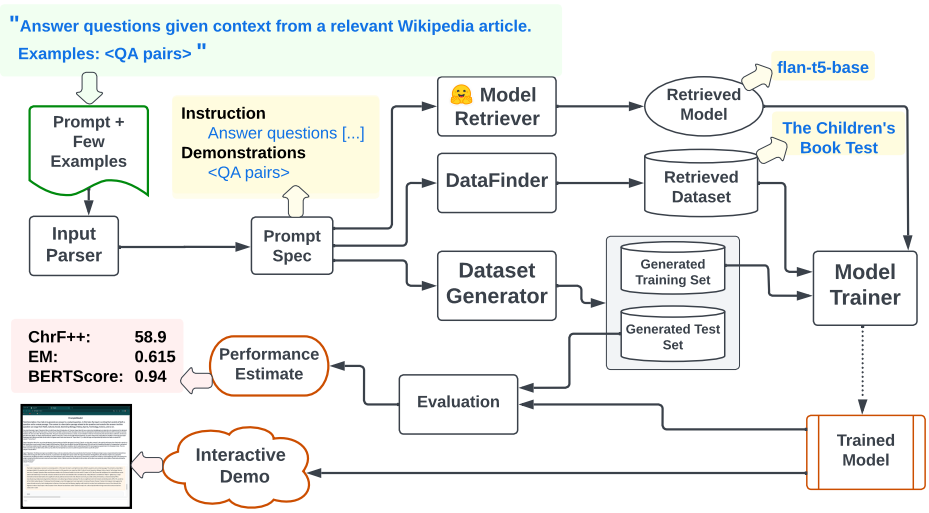

Figure 2: The Prompt2Model architecture seeks to automate the core machine learning development pipeline, allowing us to train a small yet accurate model from just a prompt.

the instruction provided is identified to be in a language other than English, we translate it to English using the DeepL API.[2]

2.2 Dataset Retriever

Given a prompt, we first try to discover existing manually-annotated data that can support a user's task description. We employ the *DataFinder* system introduced by Viswanathan et al. (2023) to retrieve datasets. By extracting user-generated dataset descriptions for each dataset in HuggingFace Datasets (Lhoest et al., 2021), we utilize DataFinder's trained bi-encoder retriever to rank the most relevant datasets. Once a relevant dataset is identified, the next step is to determine which columns of the dataset correspond to the input and the desired 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, where $k = 25$ by default, 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 columns for input and output from the dataset's schema.

2.3 Dataset Generator

Not all conceivable tasks have any existing annotated data, and many tasks are only somewhat relevant to an existing dataset. To support a wide range of tasks, we produce synthetic training data to meet the user-specific requirements parsed by the *Prompt Parser*. This task presents inherent challenges related to cost efficiency, generation speed, example diversity, and quality control. Our strategy, carefully engineered to address these challenges, comprises the following key components:

High-Diversity Few-Shot Prompting We use automated prompt engineering to generate a diverse dataset. We assemble dynamic prompts using the user's instructions and a mix of user-provided demonstration examples and previously generated examples. This strategy yields diverse prompts that effectively guide the language model to produce contextually relevant, high-quality outputs. On a test of 200 QA examples, this strategy reduced the number of duplicates from 120 to 25.

Temperature Annealing Inspired by Adam (Kingma and Ba, 2014), as the dataset expands, we adjust the sampling temperature from low (favoring deterministic outputs) to high (encouraging diverse exploration) proportionally to the number of examples already generated. This gradual temperature modulation aids in preserving output quality while simultaneously encouraging diversity.

Minimum Bayes Risk Decoding Given that LLM may generate non-unique or incorrect outputs for the same inputs, we use Minimum Bayes Risk decoding (Bickel and Doksum, 1977) to select pseudo-labels, which is a generalized form of the widely-used *self-consistency* filtering (Wang et al., 2022). Specifically, we create a consensus output for each unique input by selecting the shortest answer among the most frequent ones. This

[2] https://www.deepl.com/en/docs-api

promotes the consistency and accuracy of the generated dataset with unique input-output pairs.

Asynchronous Batching API requests are parallelized using *zeno-build* (Neubig and He, 2023). We use additional mechanisms, such as dynamic batch size and throttling, to optimize API usage.

2.4 Model Retriever

Besides training data, we also need to select an appropriate model (e.g., pre-trained on relevant domains) to finetune. To support many tasks with a unified model interface, we limit ourselves to encoder-decoder architectures on Hugging Face (Wolf et al., 2020). This restriction still leaves a large set of pretrained models to choose from, e.g. `flan-t5-base` for general-purpose NLP (Chung et al., 2022a), `Salesforce/codet5-base` for coding-related tasks (Wang et al., 2021b), `MaryaAI/opus-mt-ar-en-finetuned-ar-to-en` for Arabic-to-English translation (Tiedemann and Thottingal, 2020). We frame the problem of selecting a pretrained model as a search problem. Using the user's instruction as a query, we search against all 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 first use `gpt-3.5-turbo` to create a *hypothetical model description* given the user's instructions. We show an example of a hypothetical document generated for a question-answering instruction in Figure 3. Using this description as an expanded query, we then apply the BM25 algorithm to compute query-model similarity scores (Robertson et al., 1995). We filter out models whose size (in bytes) exceeds a user-specified threshold (set to 3GB by default). Using the intuition that highly-downloaded models are more likely to be high in quality, we finally rank all remaining models by

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

and use the top-ranked model for finetuning.

2.5 Training and Evaluation

Dataset Processing We want to train a model by leveraging two datasets- one generated and one retrieved. To sidestep the challenge of automatically making schema-specific modeling decisions (e.g. constructing different architectures for classification tasks and generation tasks), we cast all datasets as "text-to-text" generation problems (Raffel et al., 2020). We textualize the input columns of each dataset and prepend the user's instructions to the input to guide the model.

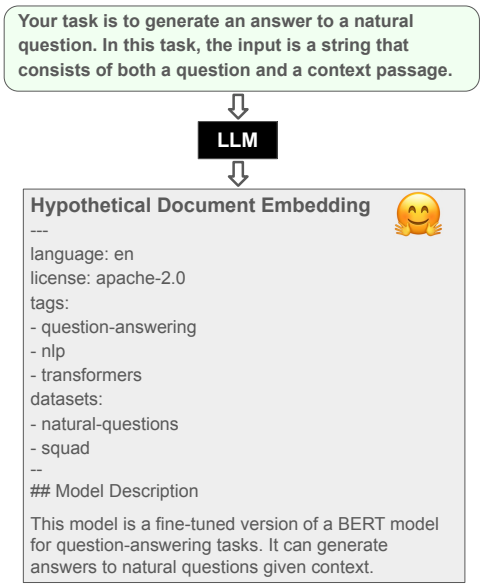

Figure 3: For our model retriever, we first construct a hypothetical model description for a query, then compute similarity scores between that hypothetical model description and the descriptions of real models.

Training Since each example in our dataset consists of an instruction followed by input text, we follow the instruction fine-tuning paradigm (Chung et al., 2022b) to train the student model. We concatenate the retrieved and generated datasets and shuffle them before training. We use the same default hyperparameters for all tasks, which we empirically find effective.[3] We train with the AdamW optimizer with `lr=5e-5` for 3 epochs, which takes roughly an hour for all tasks.

Evaluation Our *Model Evaluator* automatically evaluates models using three metrics: `Exact Match`, `ChrF++` (Popović, 2015), and `BERTScore` (Zhang et al., 2019). `ChrF++` balances precision and recall to assess text generation quality. `Exact Match` measures how often the model output perfectly matches the exact reference. `BERTScore`, comparing the model output and reference in the embedding space, captures semantic similarities despite different wordings or phrasings. We use XLM-R (Conneau et al., 2020) as the encoder for BERTScore to support multilingual evaluation.

[3] In future work, we plan on implementing automated hyperparameter selection using generated validation data.

2.6 Web App Creation

The final step in Prompt2Model is the automatic creation of a graphical user interface that allows downstream users to interact with the trained model. This web application, built using Gradio (Abid et al., 2019), can then be easily deployed publicly on a server.

3 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 first consider a common use case where pretrained models and training datasets are plentiful. We use SQuAD (Rajpurkar et al., 2016) as ground truth to evaluate this task.
- *Japanese NL-to-Code*: Code generation from Japanese-language queries is a scenario where related work exists but no annotated data or pretrained models are available. We use MCoNaLa (Wang et al., 2023) as gold evaluation data.
- *Temporal Expression Normalization*: We finally consider a task where there are no pretrained models or training datasets of any kind available. Here 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 datasests), we use real benchmark datasets here to report our pipeline's efficacy.

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` the same instruction and demonstrations provided to Prompt2Model, for fair comparison.

4 Experiment Results

4.1 Downstream performance

How effective is Prompt2Model at producing a high-quality model? In Table 1, we evaluated models produced by Prompt2Model, as well as our 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, at 250M parameters, is up to 700 times smaller than `gpt-3.5-turbo` (which is believed to contain 175B parameters).

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`, which we used to power our dataset generator. We also examine the effect of removing specific parts of our pipeline — model retrieval and dataset retrieval.

We observe that Prompt2Model's performance on MCoNaLa's Japanese-to-Python task is significantly worse than `gpt-3.5-turbo`. One explanation for this is the relatively low diversity in the generated dataset of Japanese queries; 45 of 5000 examples are different ways of saying "find the maximum value in a list of numbers". We do not observe this level of redundancy in our other datasets, 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 models found by the model retriever were trained on either on multiple language or code, but not both. The resultant models may lack the parametric knowledge to represent the Japanese inputs, Python outputs, or both.

4.2 Combining retrieved and generated datasets is powerful

Ideally, generated and retrieved data should be as close to the target domain as possible. In our experimental setting, where we deliberately choose prompts that mimic existing datasets, we can evaluate how well the model performs relative to a model trained on the same amount of data from the true dataset. We use SQuAD as a running example.[4] As our prompt is a description of the SQuAD passage-level question answering task (Figure 1), we exclude SQuAD from our retrieved datasets list. Instead, we evaluate models finetuned on:

[4]We focus on only SQuAD here because our other two tasks have less real training examples than the datasets we generate, making comparison impractical.

Method	#Train. data	Performance	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 model performance on SQuAD, using datasets produced by different modules of Prompt2Model, along with fully-manual annotation. Performance reported for all models is the exact match on the test set,[6] which reflects *the true task performance*. Cost of custom annotation is estimated from Rajpurkar et al. (2016) using their reported annotator pay rate of $9/hour 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.

1. 3k examples from the closest retrieved dataset[5]
2. 3k examples generated by Prompt2Model
3. The union of the above, which is what the full Prompt2Model pipeline uses
4. 3k examples from SQuAD (analogous to the user custom-annotating data for a task).

Table 2 shows the results across these four settings. While using retrieved or generated data causes a reduction in performance due to domain shift, the combination of the two methods achieves similar performance to using the true dataset. In the real world, where the user would need to custom-annotate data for their task, Prompt2Model allows for *similar performance at less than 1% of the cost*.

4.3 Our generated evaluation data can identify real modeling improvements

High-quality generated data should also allow us to *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 according to the generated test data and the test data from the target (real) dataset. We evaluate the Kendall's rank correlation (Kendall, 1938) between the two rankings to determine if our generated data can effectively determine which models are likely to perform well downstream. This is closely related to the concept of concurrence between benchmarks (Liu et al., 2023a); however, we are evaluating whether the generated and real data rank *specific models* in the same ordering, rather than *modeling approaches*.

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

5 Discussion and Conclusion

We propose Prompt2Model, a framework that automatically train task-specific models using only natural language prompts. Our proof-of-concept experiments show that Prompt2Model uses the same easy-to-use interface of LLMs to deliver small yet accurate models, and that its generated datasets can be used to estimate real-world performance. Besides being a ready-to-use tool, Prompt2Model's extensible design and modularized implementation 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. We hope that our platform enables future work that looks more deeply into quality assurance on the generated data and the model. Interesting questions include, how much data should we generate for downstream model training, and how diverse should it be? How do we effectively mix the retrieved and generated dataset such to achieve complementary strengths (e.g. using dataset generation to focus on the expected inputs to the model that the retrieved dataset fails to cover)? Additionally, since people often struggle to articulate their needs up front, future extensions will also need to address the challenge of human-in-the-loop correction – either by offering potential strategies to help humans iteratively refine prompts, or allowing humans to perform post-hoc fixes when the task metadata extraction and generated data do not align with their intentions. We hope to propose explicit challenges and invite the community to contribute novel implementations of various components in our framework.

[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).

[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 task.

Limitations

One of the primary limitations of our system is that our current experiments have all been conducted using the `gpt-3.5-turbo` API (used for prompt parsing, dataset generation, and model retrieval). This proprietary LLM is paid and closed-source, which makes this problematic as a scientific artifact (Rogers et al., 2023). Furthermore, the service provider of this LLM, OpenAI, prohibits the use of their API to create models that may compete with OpenAI, creating potential legal concerns with the use of Prompt2Model in commercial applications. Our tool has been designed to be extensible, and we are exploring the integration of open-source LLMs as a way to avoid our 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. This model is believed to be similar to GPT-3 (Brown et al., 2020), which was trained on a corpus consisting of 93% English documents, 1% German documents, 1% French documents, and <5% documents in any other language. Subsequently, our work may exacerbate existing disparities in language technologies between high-resource languages and low-resource languages.

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 HuggingFace may lessen the likelihood of a downstream model replicating harms from the prompted model's outputs, though more investigation is needed here. 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 innovations made by the community; particularly, to people who would use NLP models downstream but may not have the domain-specific knowledge to select data, models, and hyperparameters themselves. Prompt2Model may also prove useful for early NLP researchers by providing a starting point for intuitions about baselines for various tasks and the similarities between a described task and existing community 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.

Peter J. Bickel and Kjell A. Doksum. 1977. Mathematical statistics: Basic ideas and selected topics.

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*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022a. Scaling instruction-finetuned language models.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022b. Scaling instruction-finetuned language models.

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.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

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.

Graham Neubig and Zhiwei He. 2023. Zeno GPT Machine Translation Report.

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.

Emre Sezgin, Joseph Sirrianni, and Simon L. Linwood. 2022. Operationalizing and implementing pretrained large ai linguistic models in the united states healthcare system: An outlook of gpt-3 as a service. *JMIR Medical Informatics*, 10(2).

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.

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. 2021a. 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.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021b. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *EMNLP*.

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.